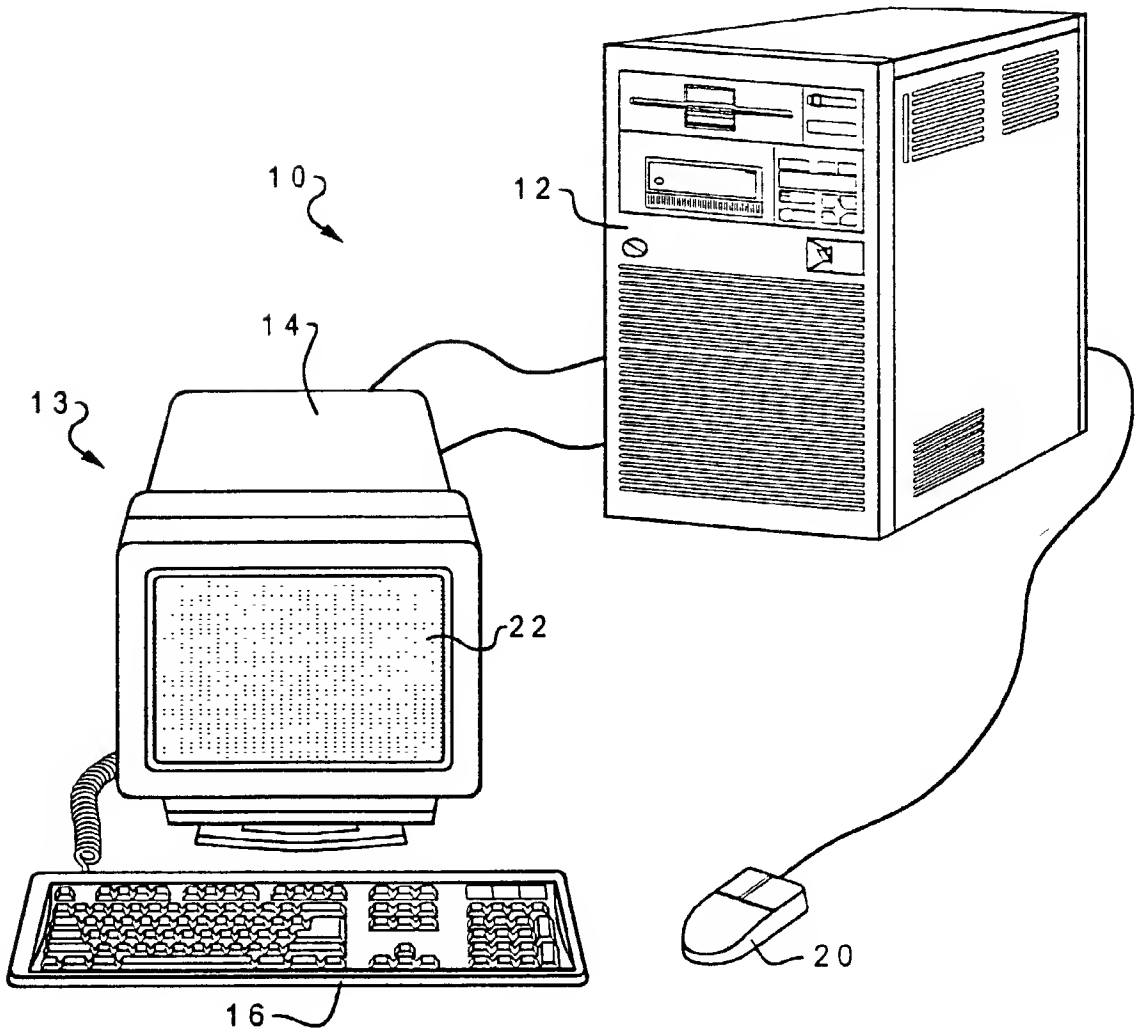


Fig. 1

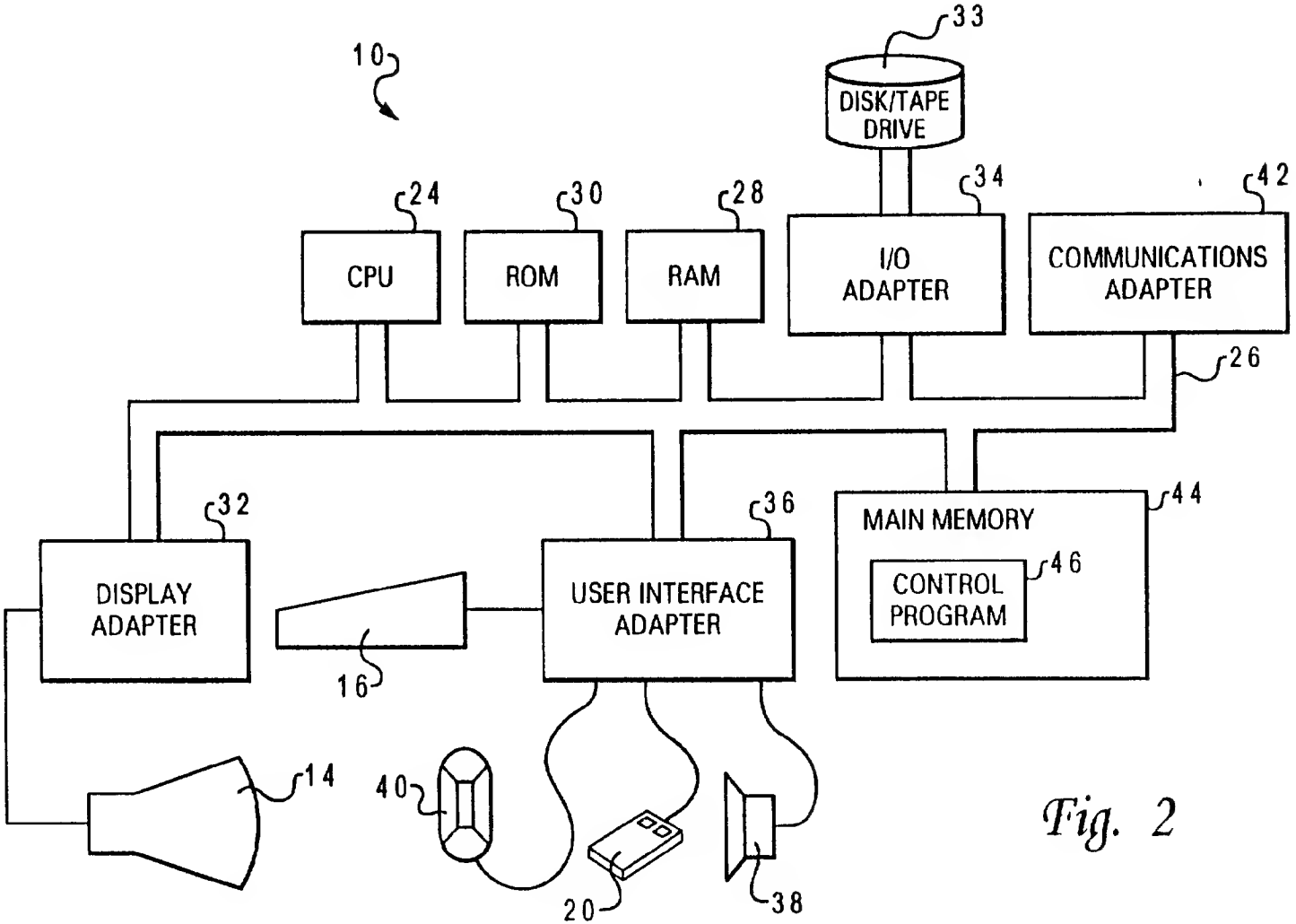


Fig. 2

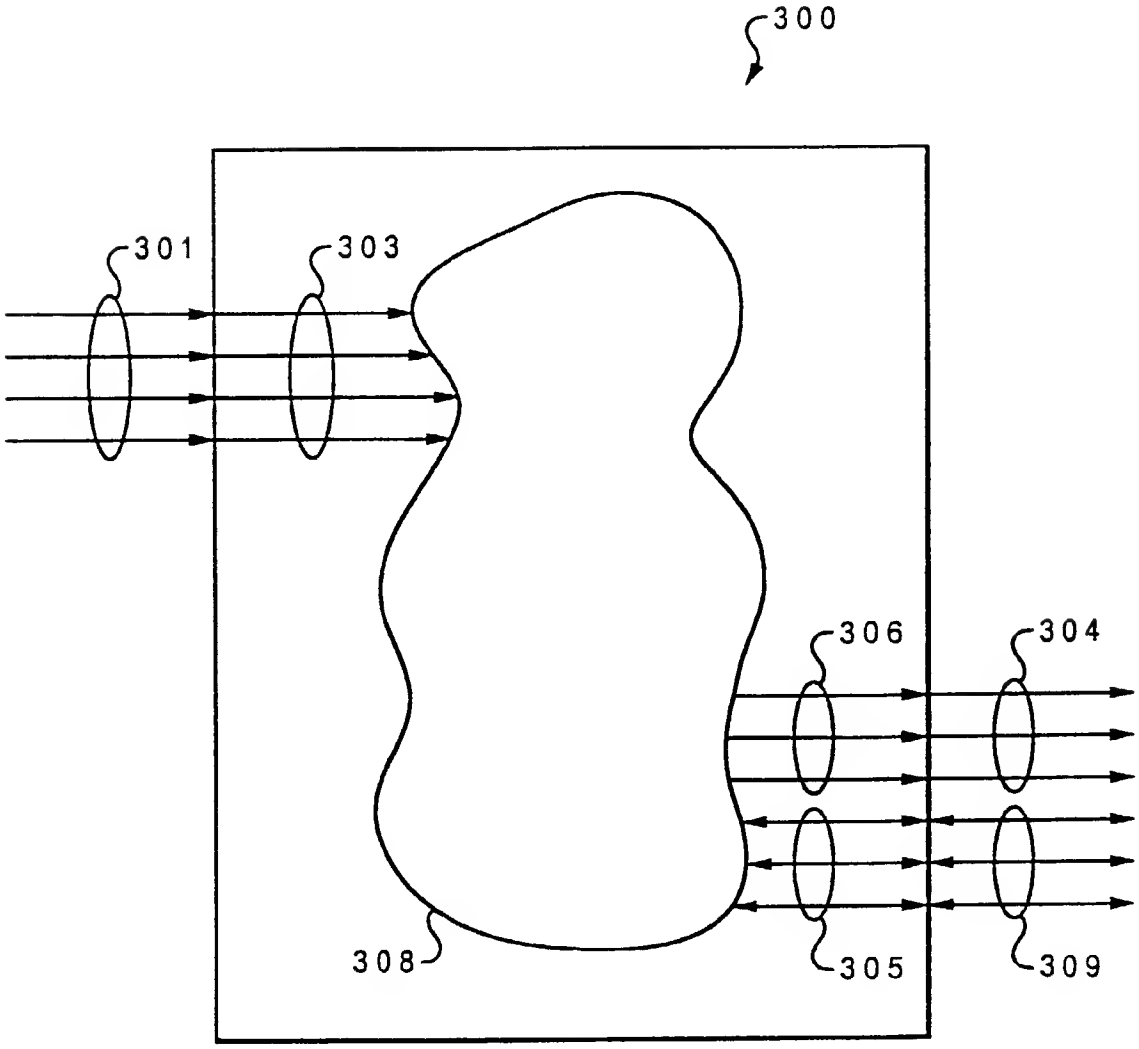


Fig. 3A

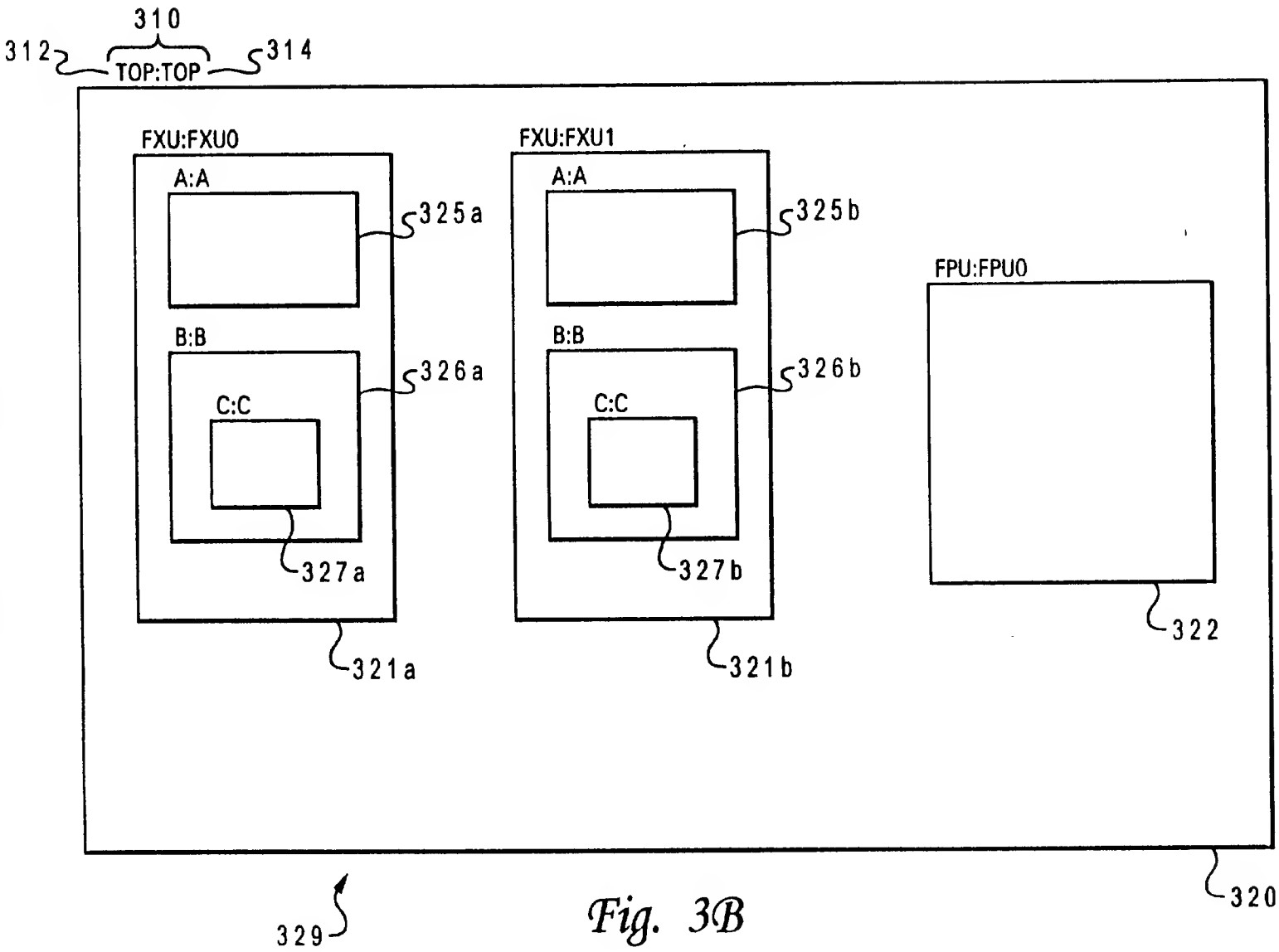

310
312
TOP:TOP
314
FXU:FXU0
A:A
325a
B:B
326a
C:C
327a
321a
FXU:FXU1
A:A
325b
B:B
326b
C:C
327b
321b
FPU:FPU0
322
320
329

Fig. 3B

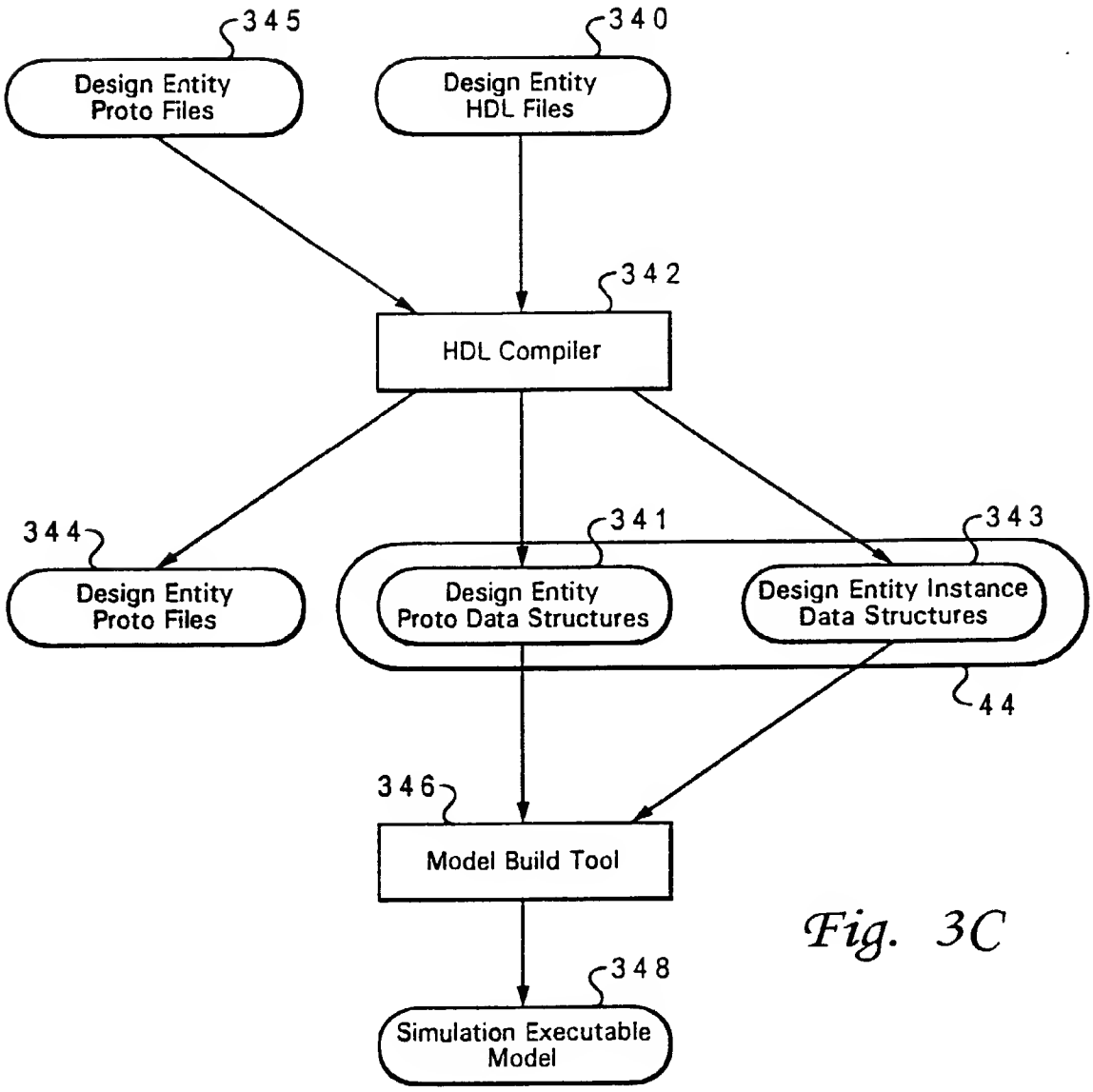

345
Design Entity Proto Files
340
Design Entity HDL Files
342
HDL Compiler
344
Design Entity Proto Files
341
Design Entity Proto Data Structures
343
Design Entity Instance Data Structures
44
346
Model Build Tool
348
Simulation Executable Model

Fig. 3C

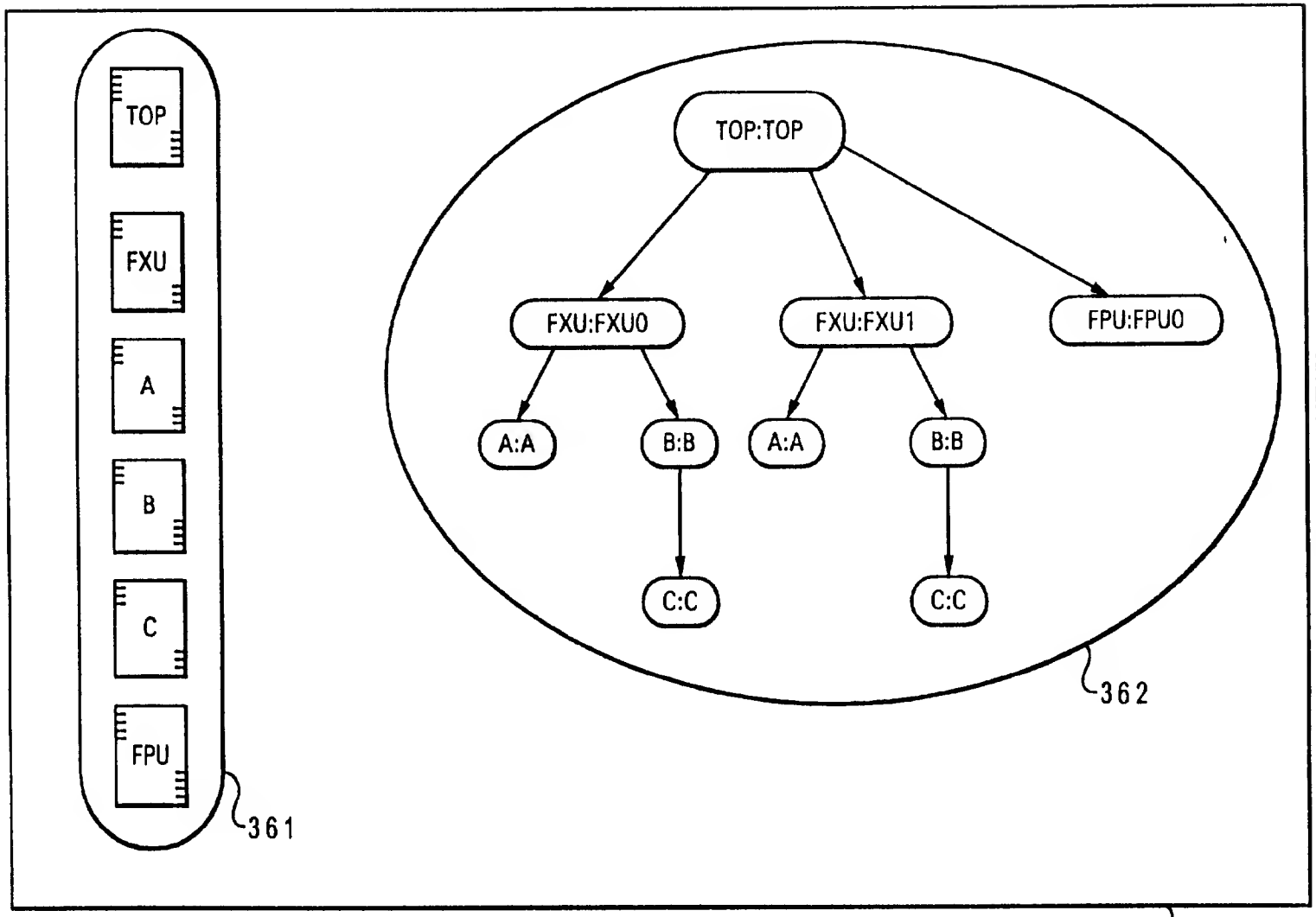

TOP
FXU
A
B
C
FPU
361
TOP:TOP
FXU:FXU0
FXU:FXU1
FPU:FPU0
A:A
B:B
A:A
B:B
C:C
C:C
362
44

Fig. 3D

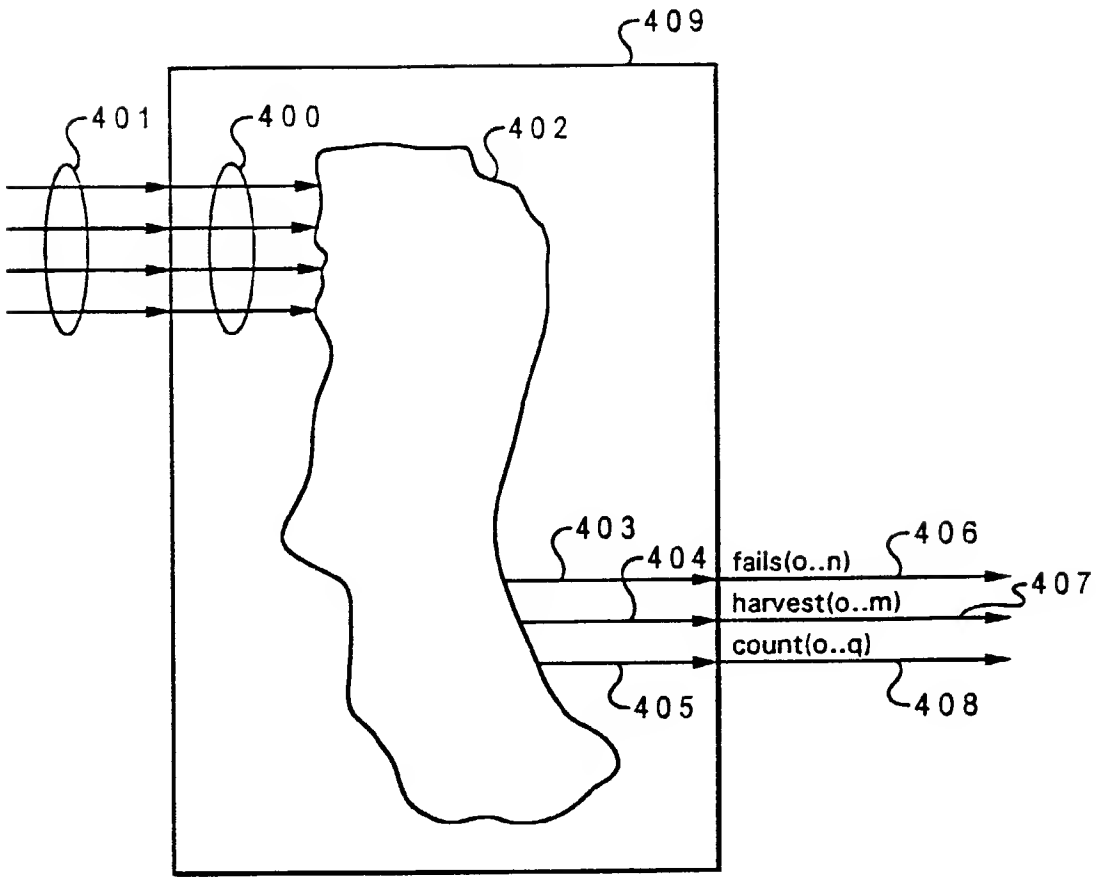

409
401
400
402
403
404
fails(o..n)
406
407
harvest(o..m)
count(o..q)
405
408

Fig. 4A

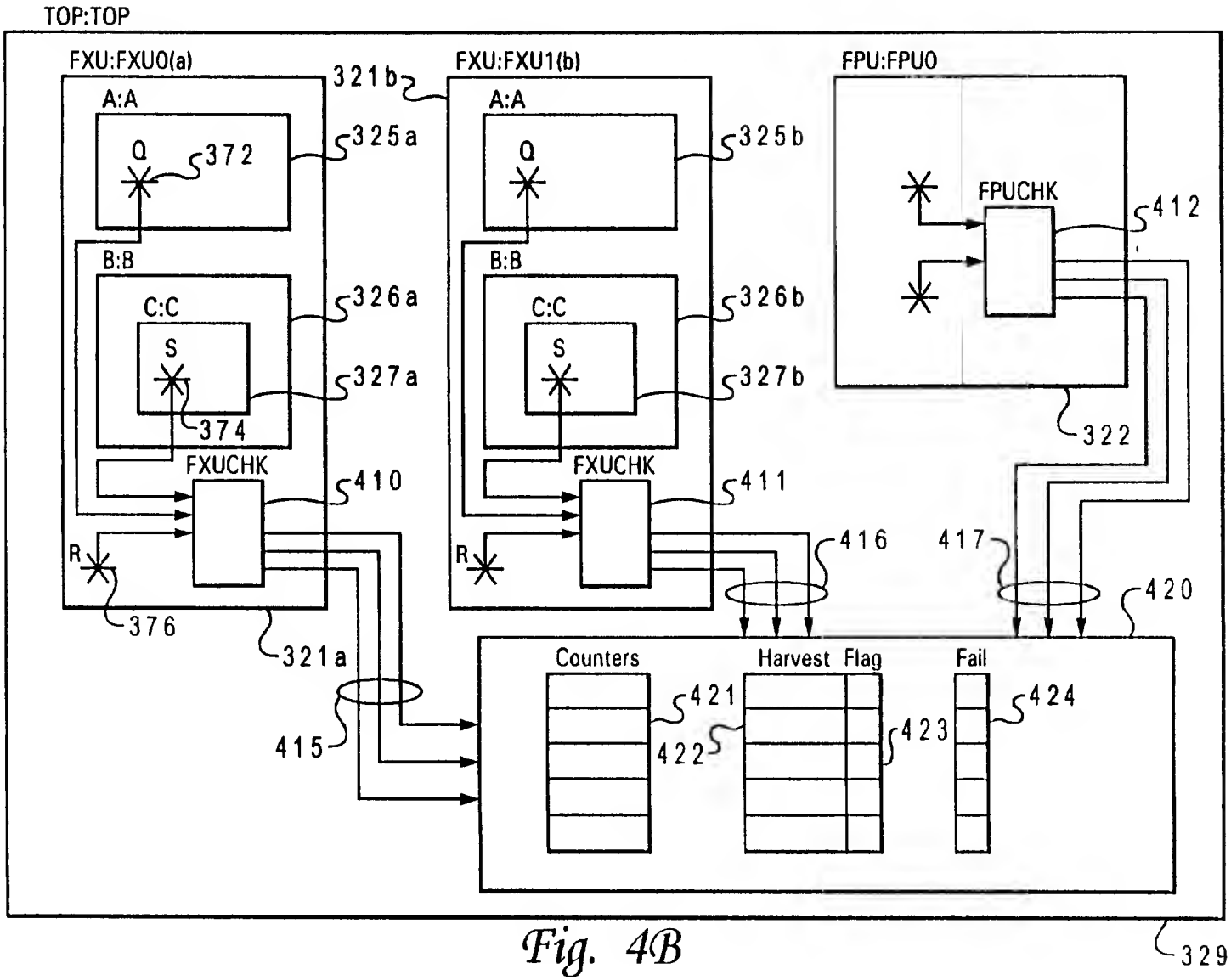

TOP:TOP
FXU:FXU0(a)
321b
FXU:FXU1(b)
FPU:FPU0
A:A
A:A
Q
372
325a
Q
325b
FPUCHK
412
B:B
B:B
C:C
326a
C:C
326b
S
S
327a
327b
374
322
FXUCHK
410
FXUCHK
411
R
R
416
417
420
376
321a
Counters
Harvest
Flag
Fail
421
424
415
422
423
329

Fig. 4B

```
ENTITY FXUCHK IS                                                    } 450

    PORT(   S_IN        :       IN std_ulogic;
            Q_IN        :       IN std_ulogic;
            R_IN        :       IN std_ulogic;
            clock       :       IN std_ulogic;
            fails       :       OUT std_ulogic_vector(0 to 1);
            counts      :       OUT std_ulogic_vector(0 to 2);
            harvests    :       OUT std_ulogic_vector(0 to 1);
        );

452 { --!! BEGIN
      --!! Design Entity: FXU;

453 { --!! Inputs
      --!! S_IN        =>      B.C.S;
      --!! Q_IN        =>      A.Q;
      --!! R_IN        =>      R;
      --!! CLOCK       =>      clock;
      --!! End Inputs

454 { --!! Fail Outputs;
      --!! 0 : "Fail message for failure event 0";
      --!! 1 : "Fail message for failure event 1";
      --!! End Fail Outputs;

455 { --!! Count Outputs;
      --!! 0 : <event0> clock;
      --!! 1 : <event1> clock;
      --!! 2 : <event2> clock;
      --!! End Count Outputs;

456 { --!! Harvest Outputs;
      --!! 0 : "Message for harvest event 0";
      --!! 1 : "Message for harvest event 1";
      --!! End Harvest Outputs;

457 { --!! End;                                                      } 451

ARCHITECTURE example of FXUCHK IS

BEGIN                                                               } 458

    ... HDL code for entity body section ...

END;                                                                                } 440
```

Fig. 4C

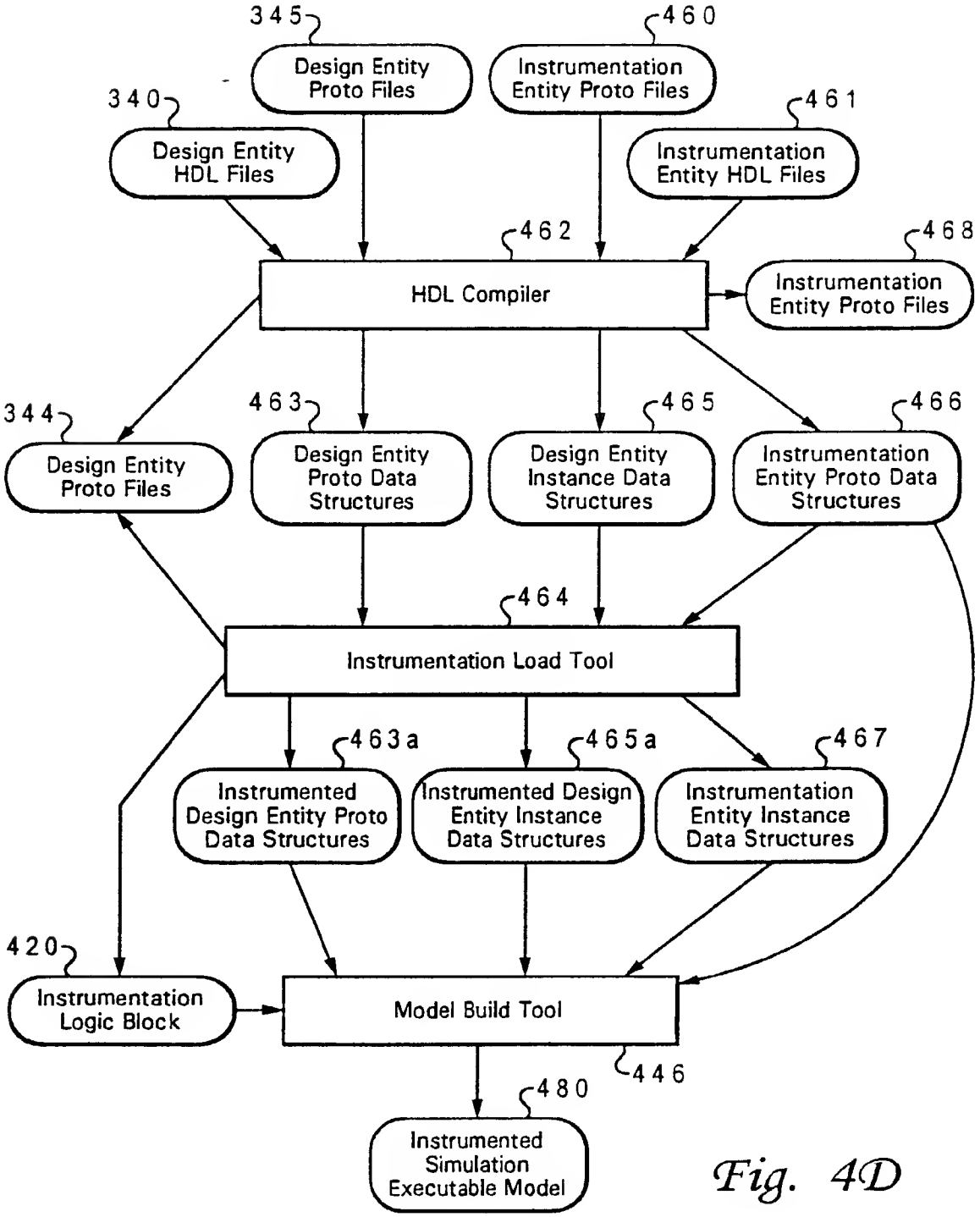

345
Design Entity Proto Files
460
Instrumentation Entity Proto Files
340
Design Entity HDL Files
461
Instrumentation Entity HDL Files
462
HDL Compiler
468
Instrumentation Entity Proto Files
344
Design Entity Proto Files
463
Design Entity Proto Data Structures
465
Design Entity Instance Data Structures
466
Instrumentation Entity Proto Data Structures
464
Instrumentation Load Tool
463a
Instrumented Design Entity Proto Data Structures
465a
Instrumented Design Entity Instance Data Structures
467
Instrumentation Entity Instance Data Structures
420
Instrumentation Logic Block
Model Build Tool
446
480
Instrumented Simulation Executable Model

Fig. 4D

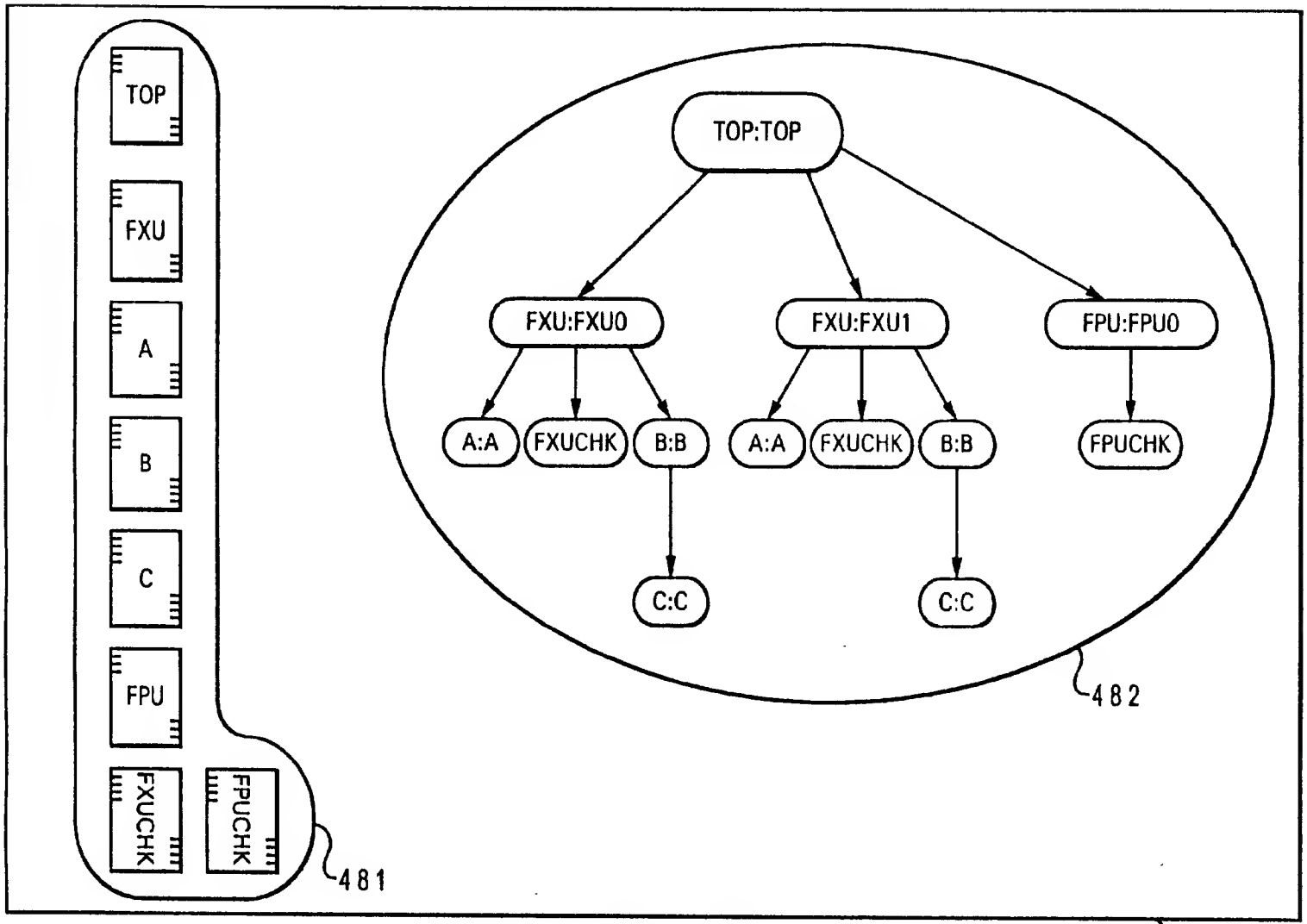

TOP
FXU
A
B
C
FPU
FXUCHK
FPUCHK
481
TOP:TOP
FXU:FXU0
FXU:FXU1
FPU:FPU0
A:A
FXUCHK
B:B
A:A
FXUCHK
B:B
FPUCHK
C:C
C:C
482
44

Fig. 4E

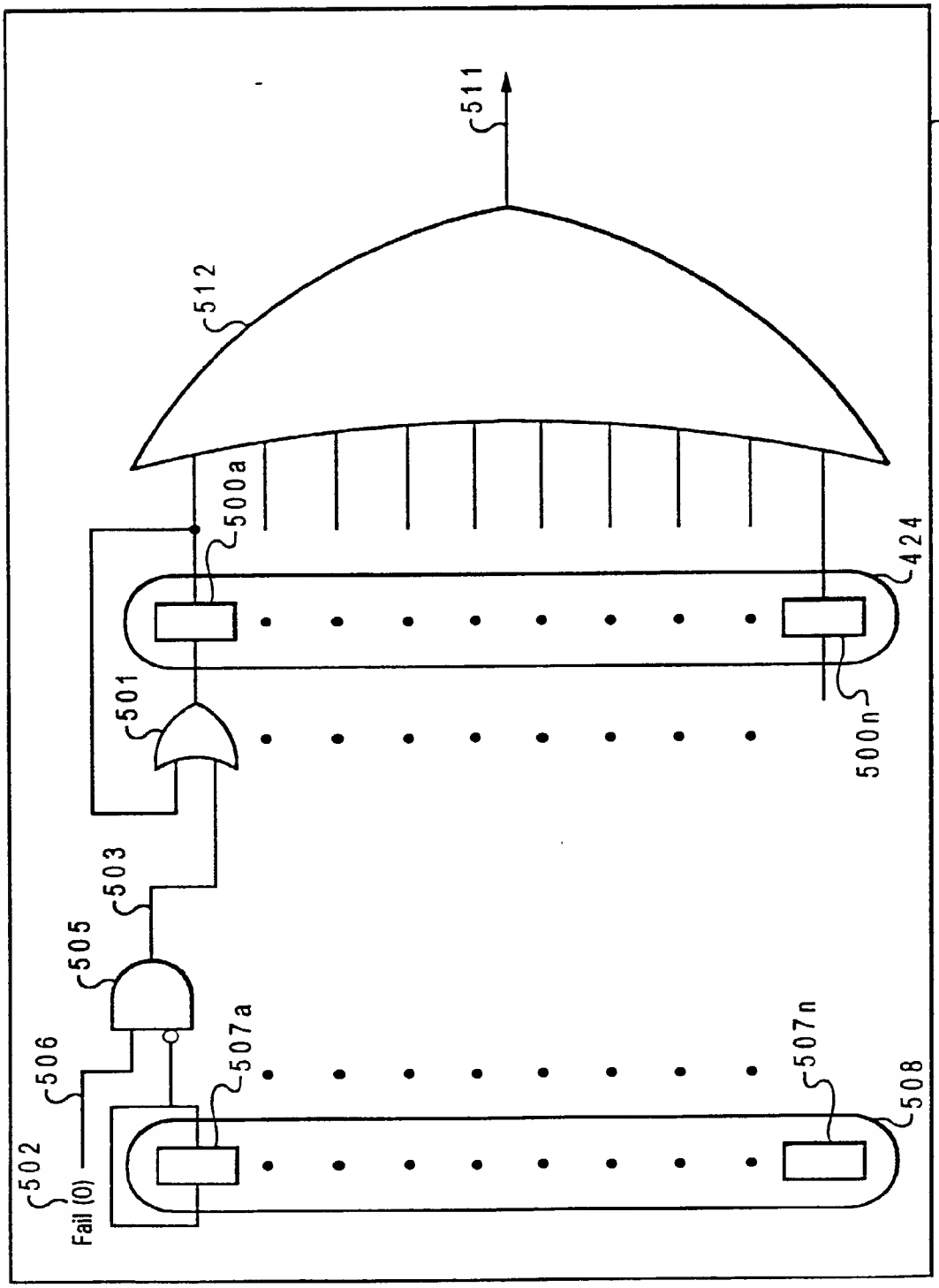

420
511
512
500a
424
501
500n
503
505
506
507a
507n
508
502
Fail (0)

Fig. 5A

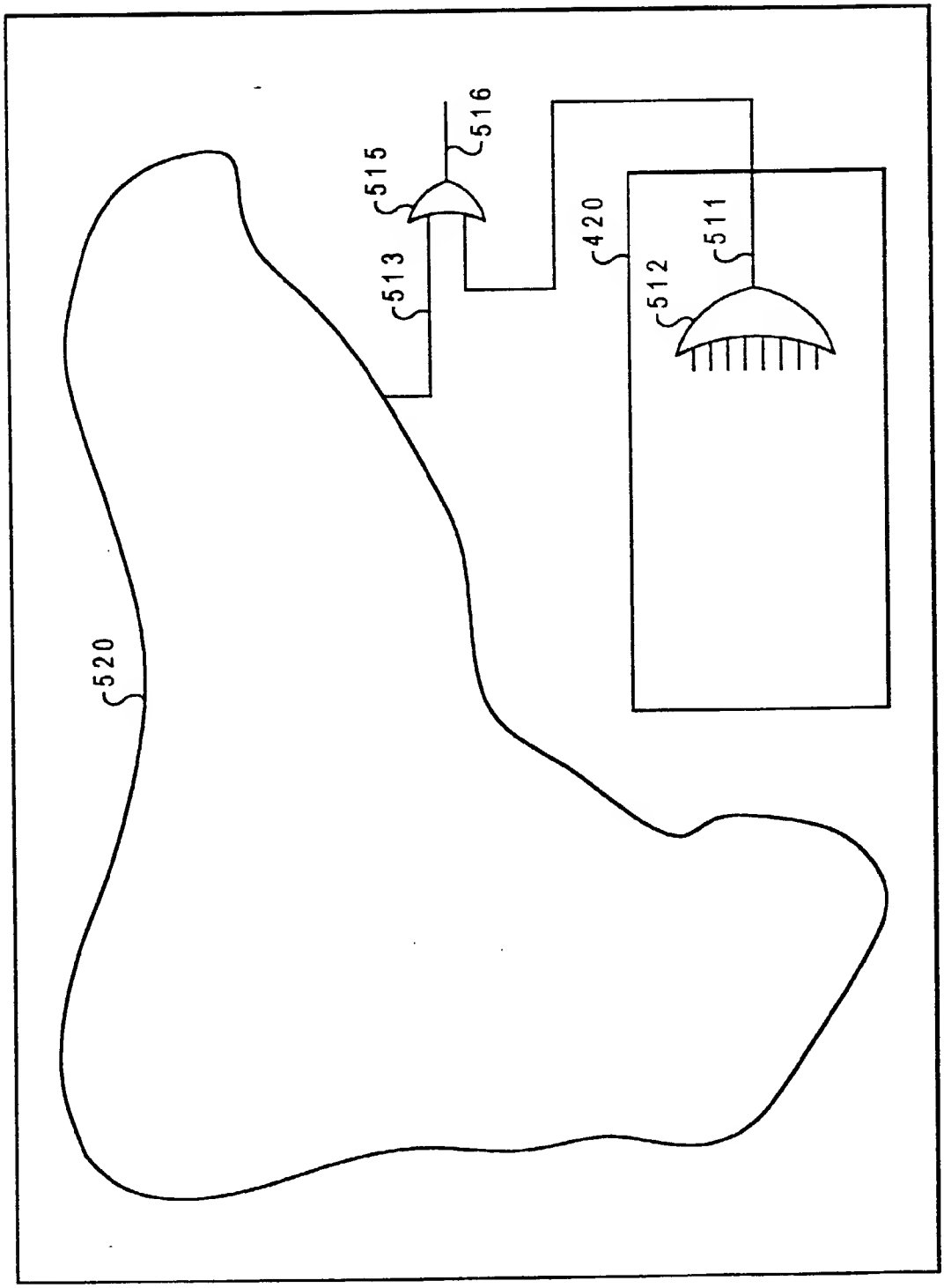


Fig. 5B

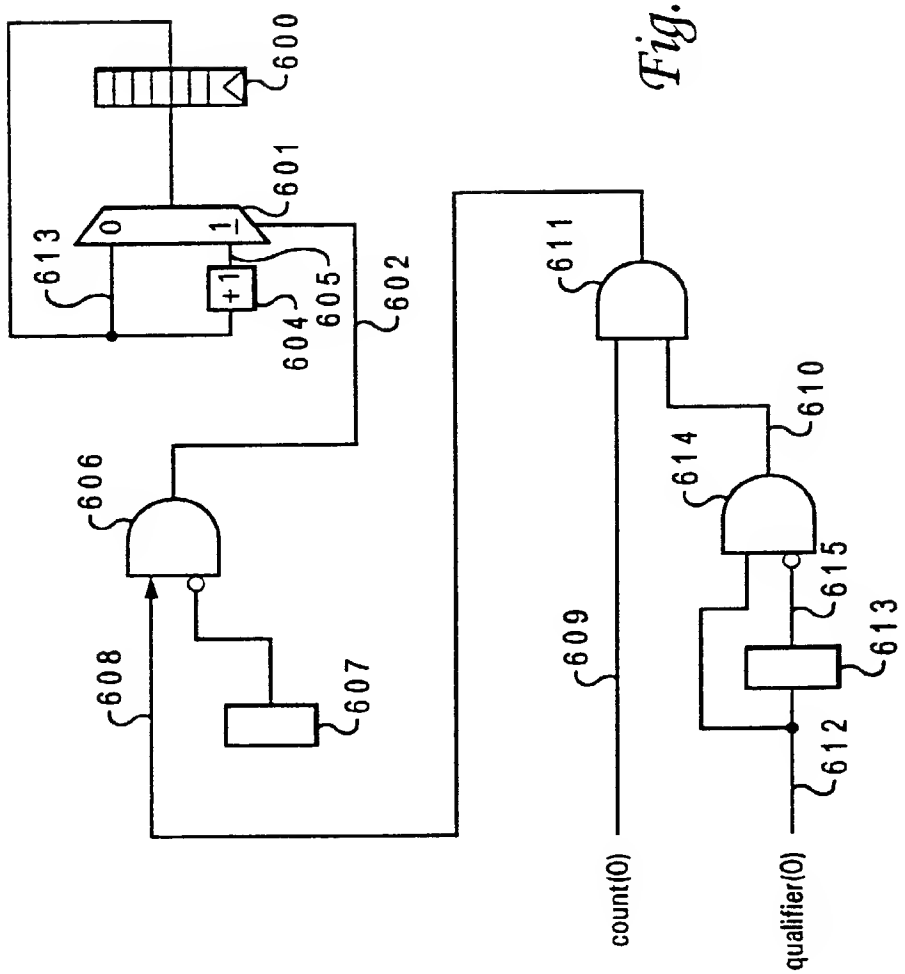

600
601
613
0
1
+1
604
605
602
606
608
607
611
609
614
610
615
613
612
count(0)
qualifier(0)

Fig. 6A

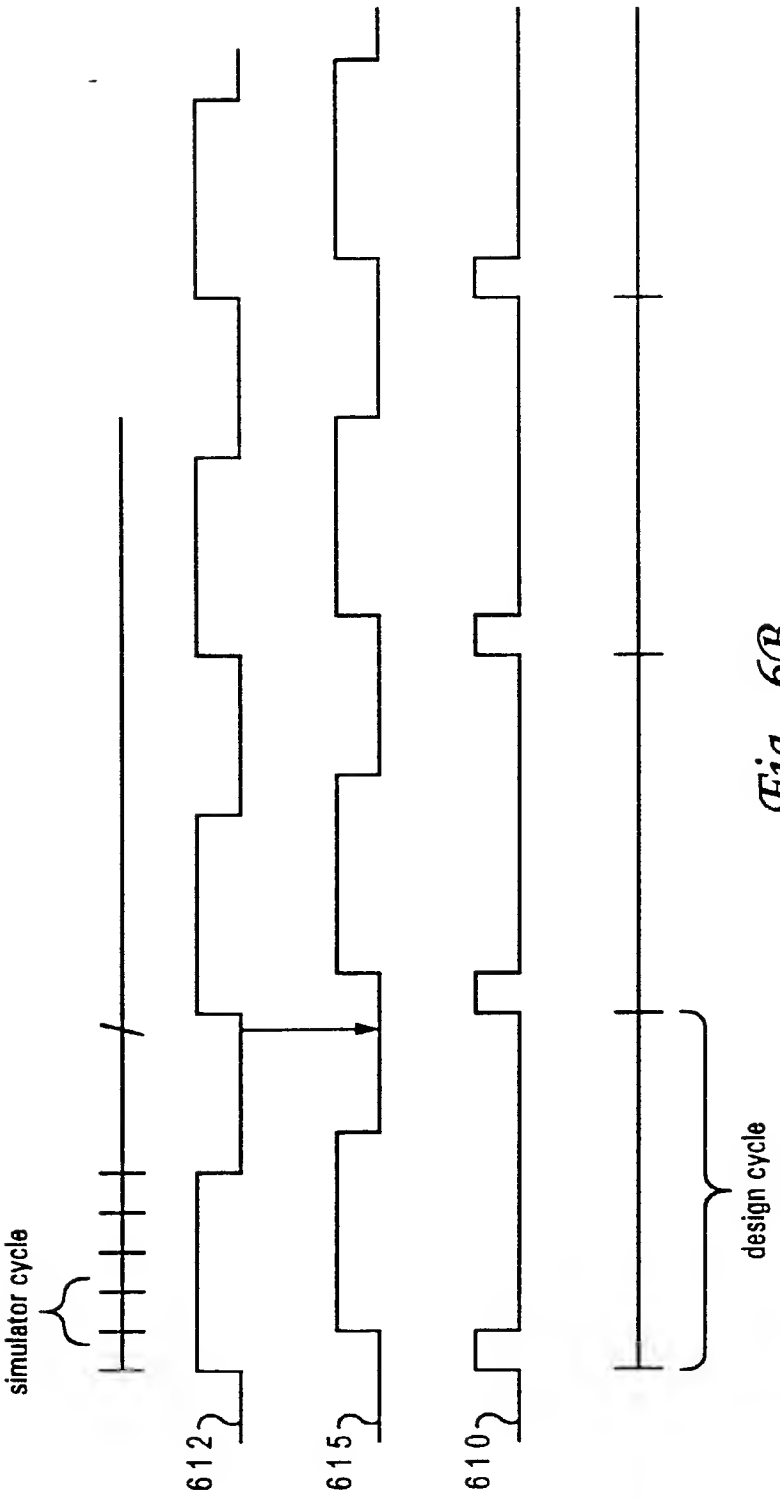


Fig. 6B

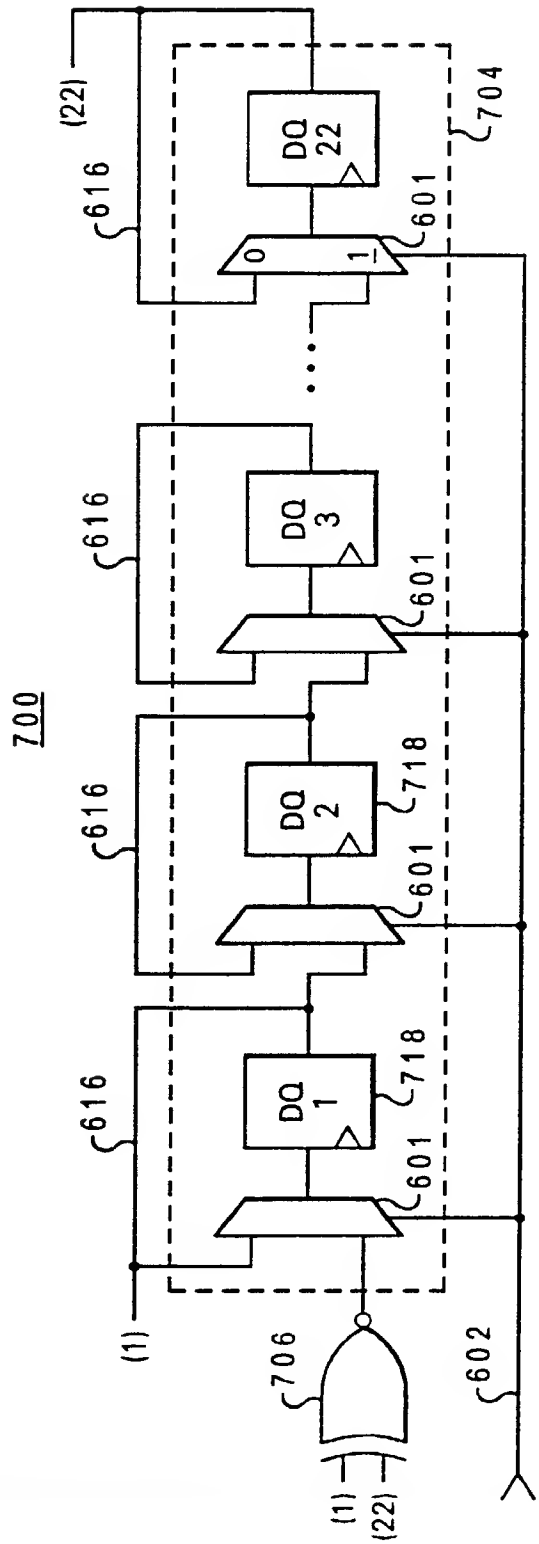


Fig. 7

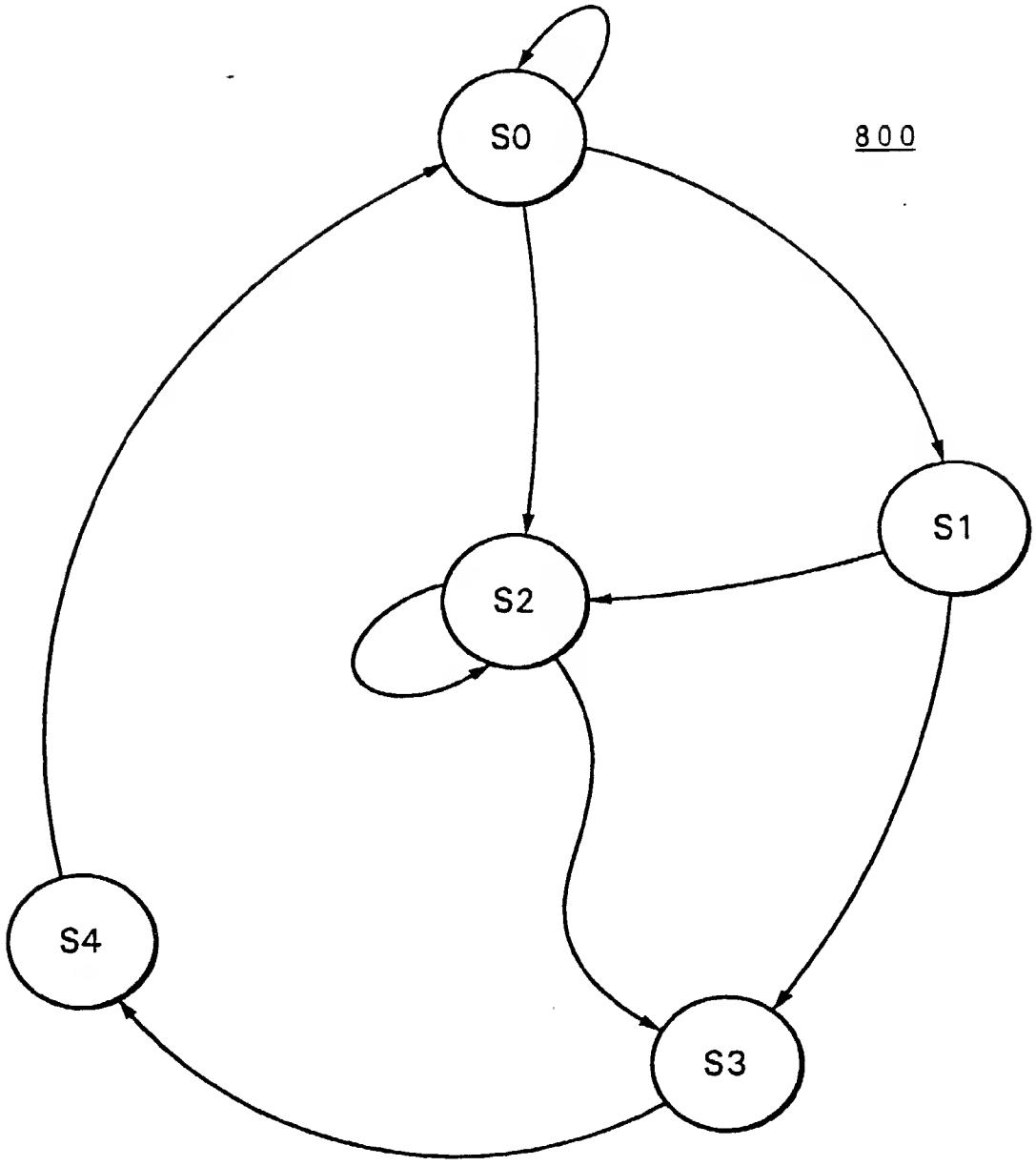


Fig. 8A
Prior Art

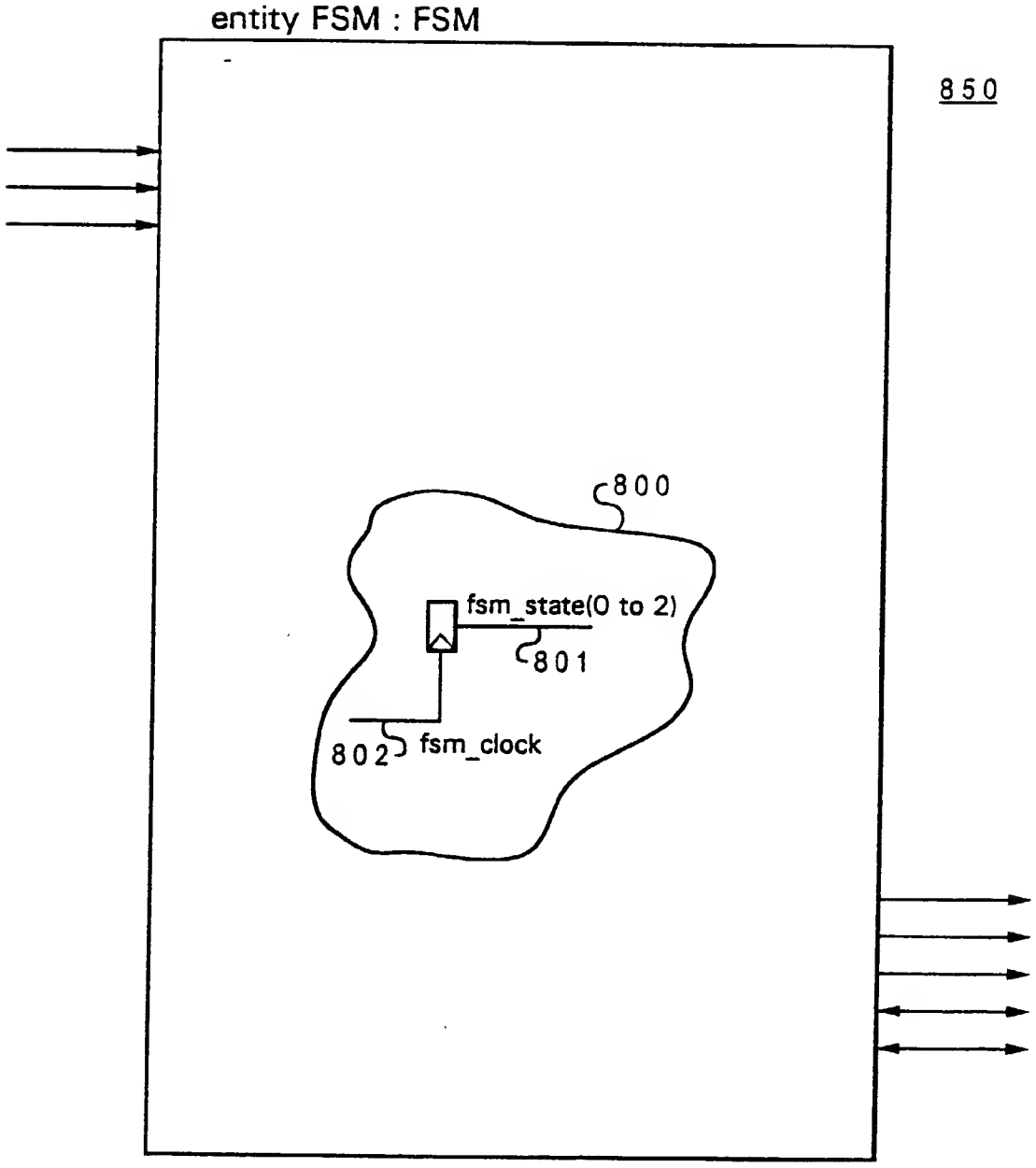


Fig. 8B
Prior Art

```
ENTITY FSM IS

    PORT(
            ....ports for entity fsm....
        );

    ARCHITECTURE FSM OF FSM IS

    BEGIN

            ... HDL code for FSM and rest of the entity ...

            fsm_state(0 to 2) < = ... Signal 801 ...

853 {  --!! Embedded FSM : examplefsm;
859 {  --!! clock          : (fsm_clock);
854 {  --!! state_vector   : (fsm_state(0 to 2));
855 {  --!! states         : (S0, S1, S2, S3, S4);
856 {  --!! state_encoding : ('000', '001', '010', '011', '100');
       --!! arcs           : (S0 = > S0, S0 = > S1, S0 = > S2,
857 {  --!!                  (S1 = > S2, S1 = > S3, S2 = > S2,
       --!!                  (S2 = > S3, S3 = > S4, S4 = > S0);
858 {  --!! End FSM;

                    ,
                    ,
                    ,
                    ,
                    ,
                    ,
                    ,

END;
```

852 } 860

Fig. 8C

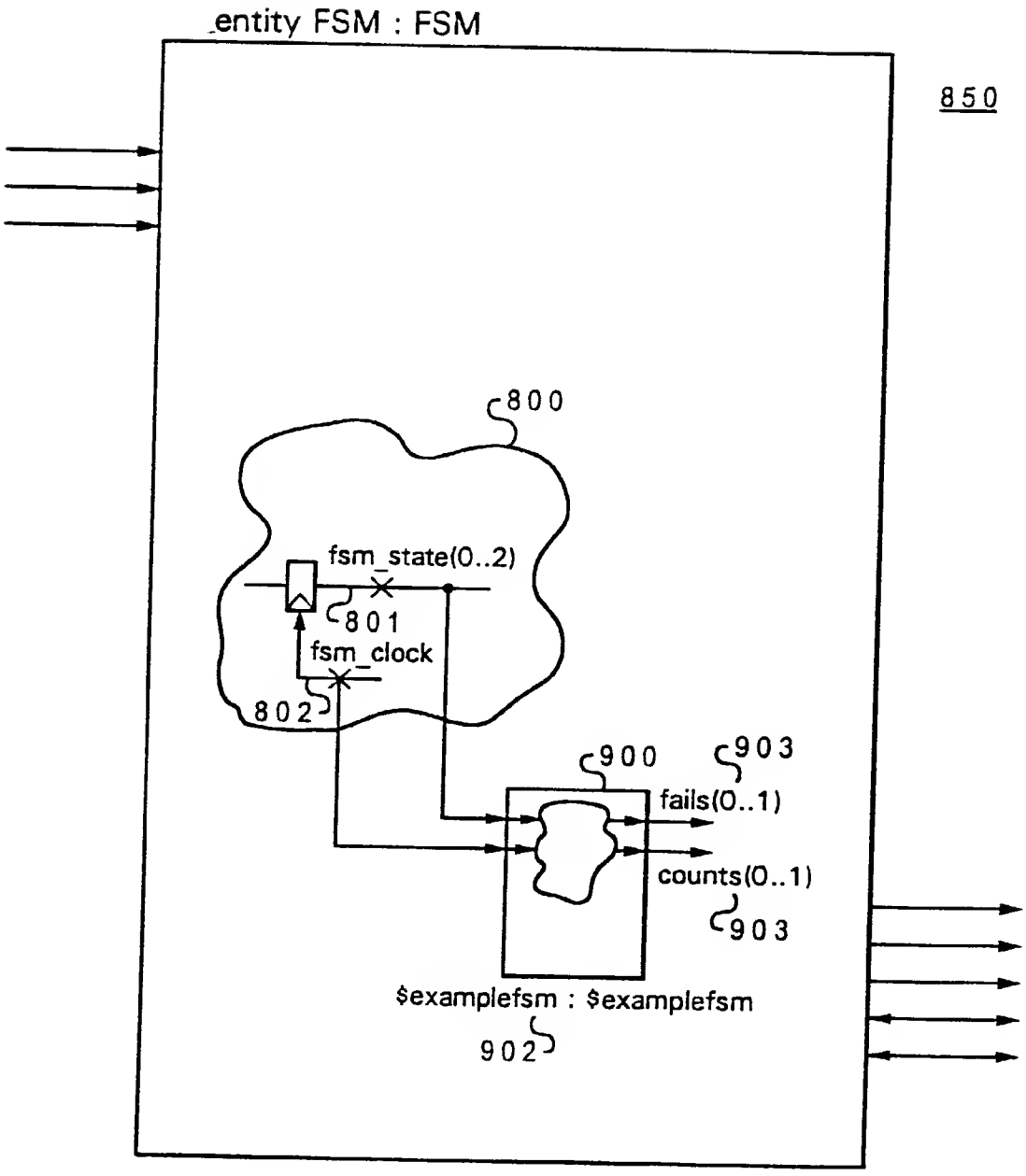

entity FSM : FSM
850
800
fsm_state(0..2)
801
fsm_clock
802
900
903
fails(0..1)
counts(0..1)
903
$examplefsm : $examplefsm
902

Fig. 9

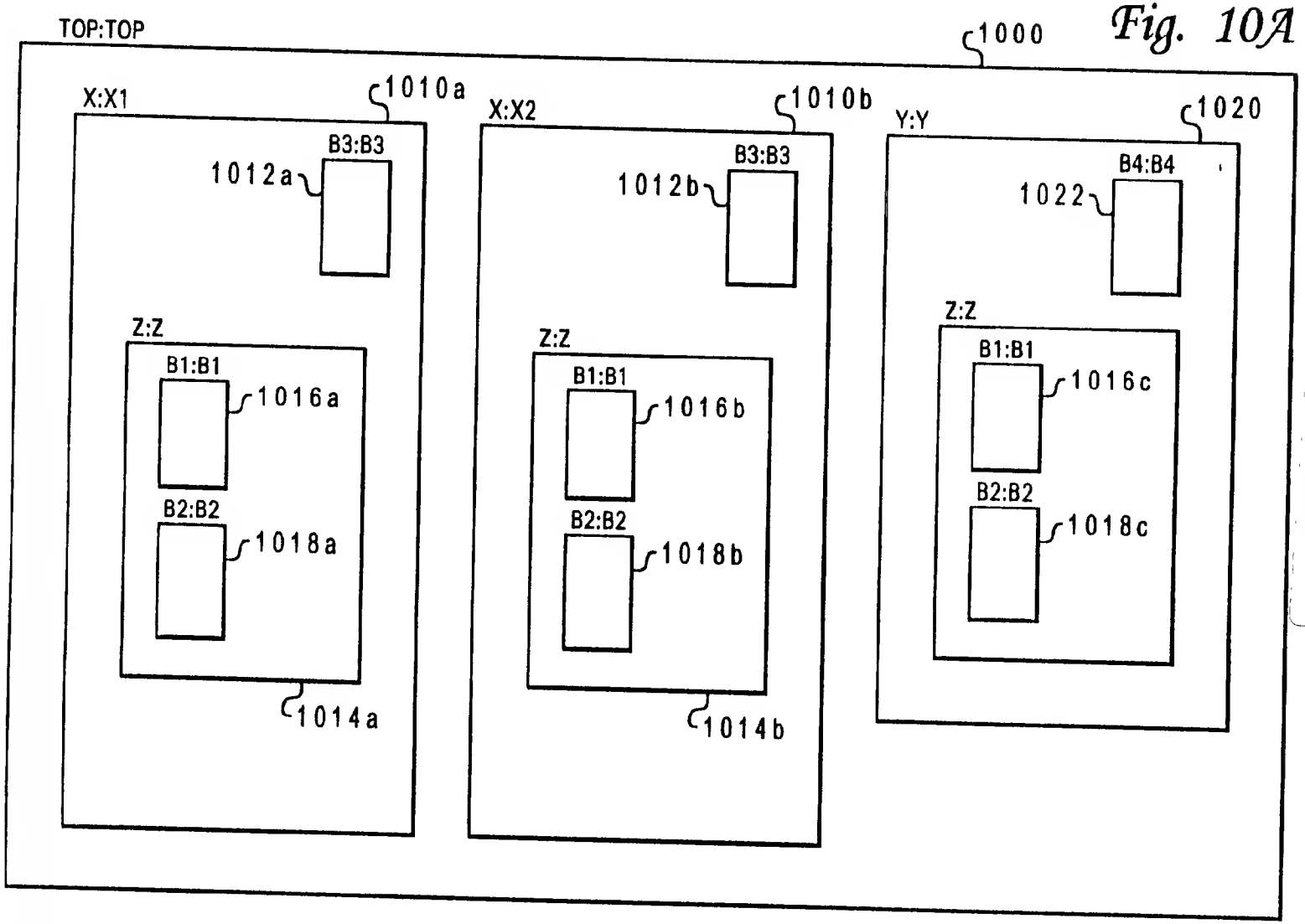

TOP:TOP
1000
X:X1
1010a
B3:B3
1012a
Z:Z
B1:B1
1016a
B2:B2
1018a
1014a
X:X2
1010b
B3:B3
1012b
Z:Z
B1:B1
1016b
B2:B2
1018b
1014b
Y:Y
1020
B4:B4
1022
Z:Z
B1:B1
1016c
B2:B2
1018c

Fig. 10A

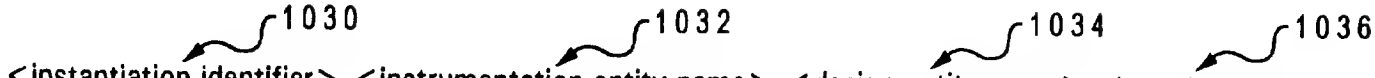


Fig. 10B

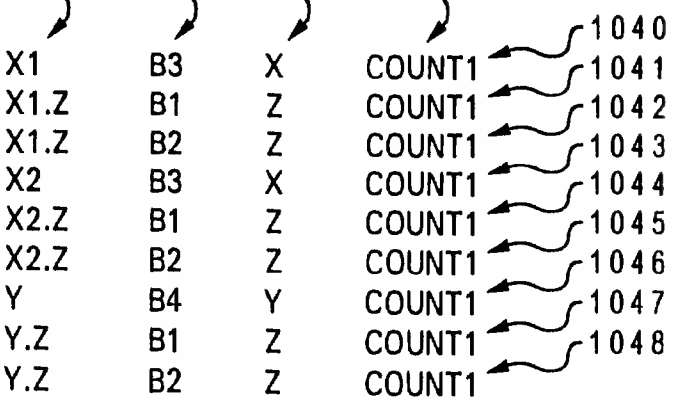


Fig. 10C

Fig. 10D

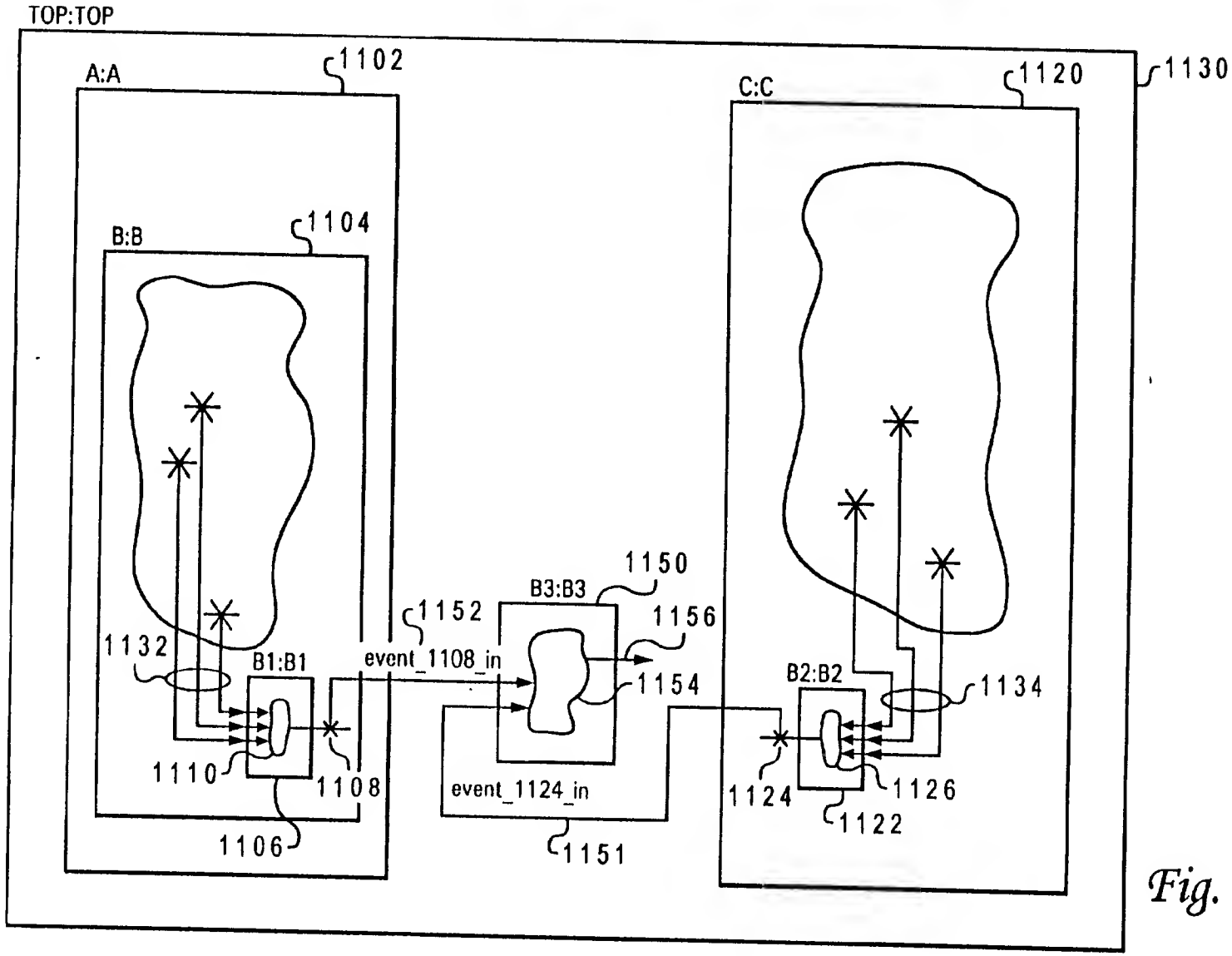

TOP:TOP
A:A
1102
1130
C:C
1120
B:B
1104
1150
B3:B3
1152
1156
1132
B1:B1
event_1108_in
1154
B2:B2
1134
1110
1108
event_1124_in
1124
1126
1122
1106
1151

Fig. 11A

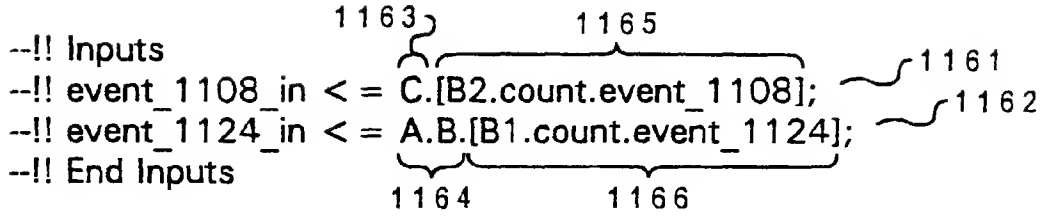


Fig. 11B

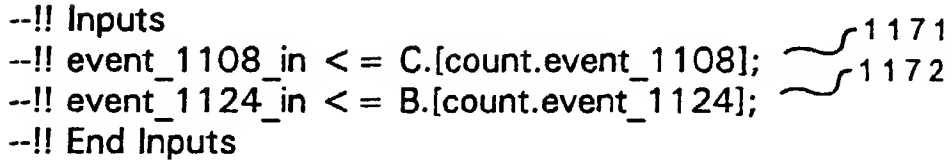


Fig. 11C

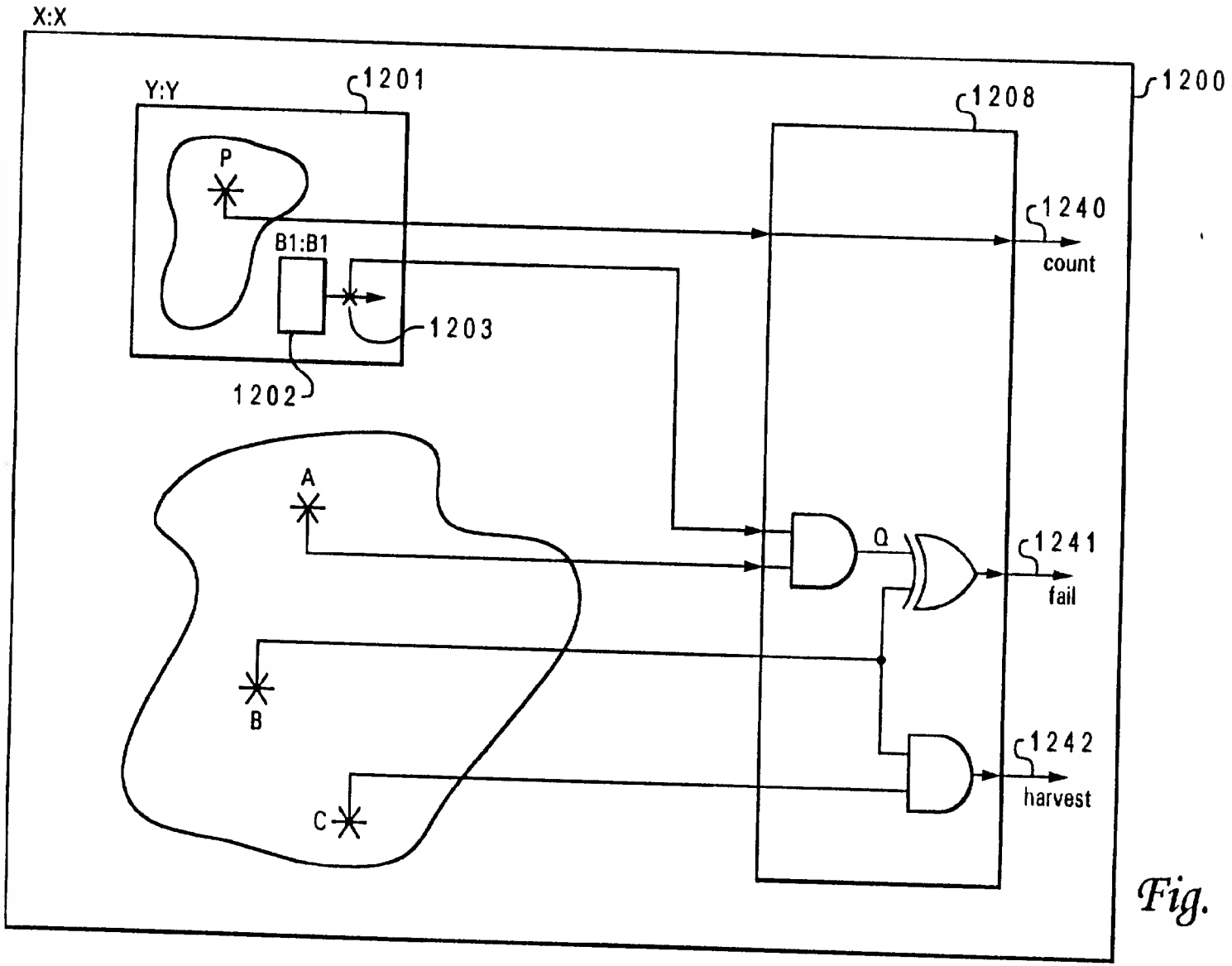

X:X
Y:Y
1201
1200
1208
P
1240
count
B1:B1
1203
1202
A
Q
1241
fail
B
1242
harvest
C

Fig. 12A

```
ENTITY X IS
     PORT(    :
              :
              :
          );

ARCHITECTURE example of X IS

BEGIN
     .
     .
     .
     .
     .
   ... HDL code for X ...
     .
     .
     .
     .
     .
     Y:Y
1221 PORT MAP(   :
                 :
             );

     A < = ....
1222 B < = ....
     C < = ....
                                                    1230
     –!! [count, countname0, clock] < = Y.P;        1232
     –!! Q < = Y. [B1.count.count1] AND A;          1234
1223 –!! [fail, failname0, "fail msg"] < = Q XOR B;
     –!! [harvest, harvestname0, "harvest msg"] < = B AND C;
END;                                          1236
```

1220

Fig. 12B

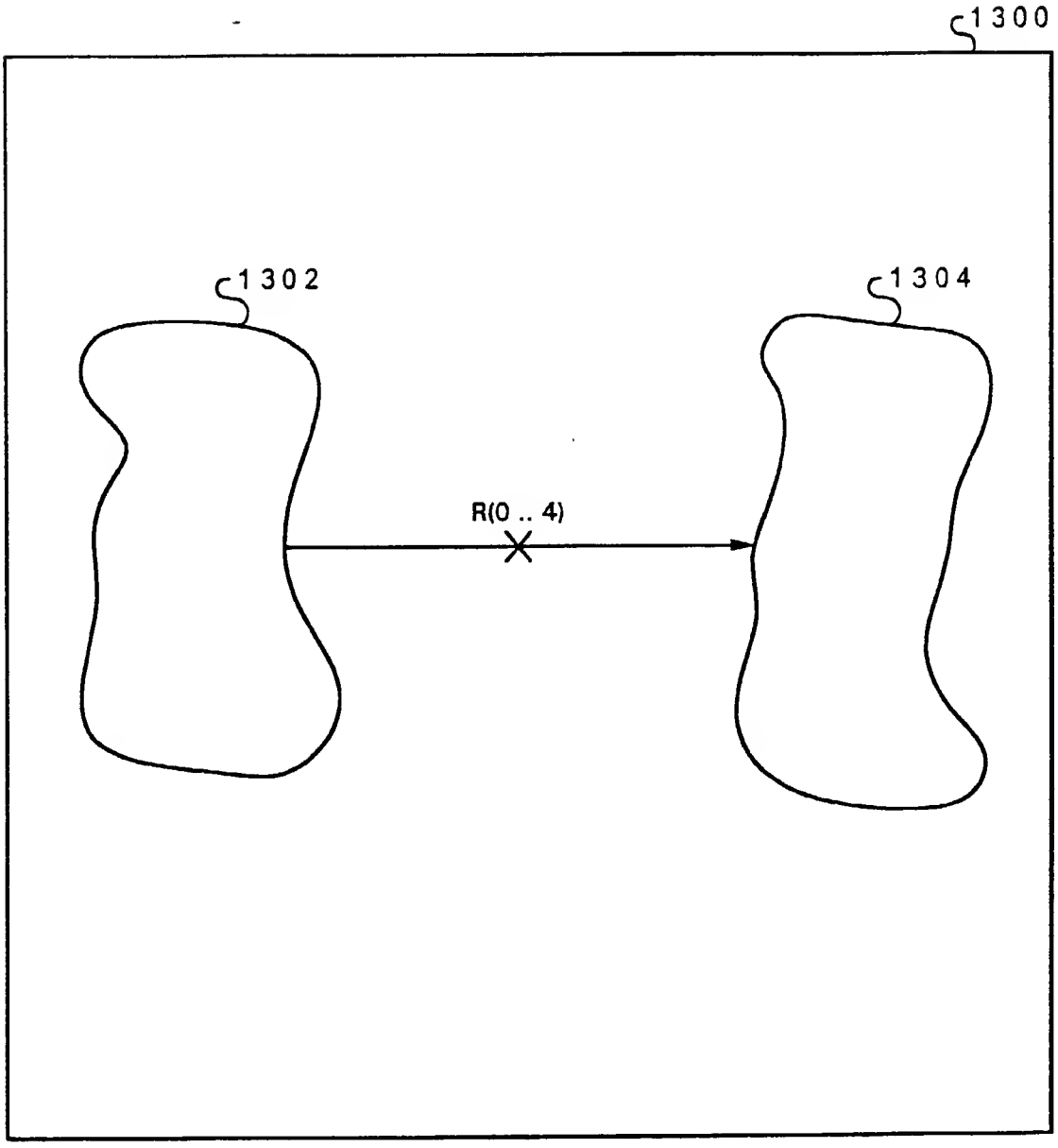


Fig. 13A

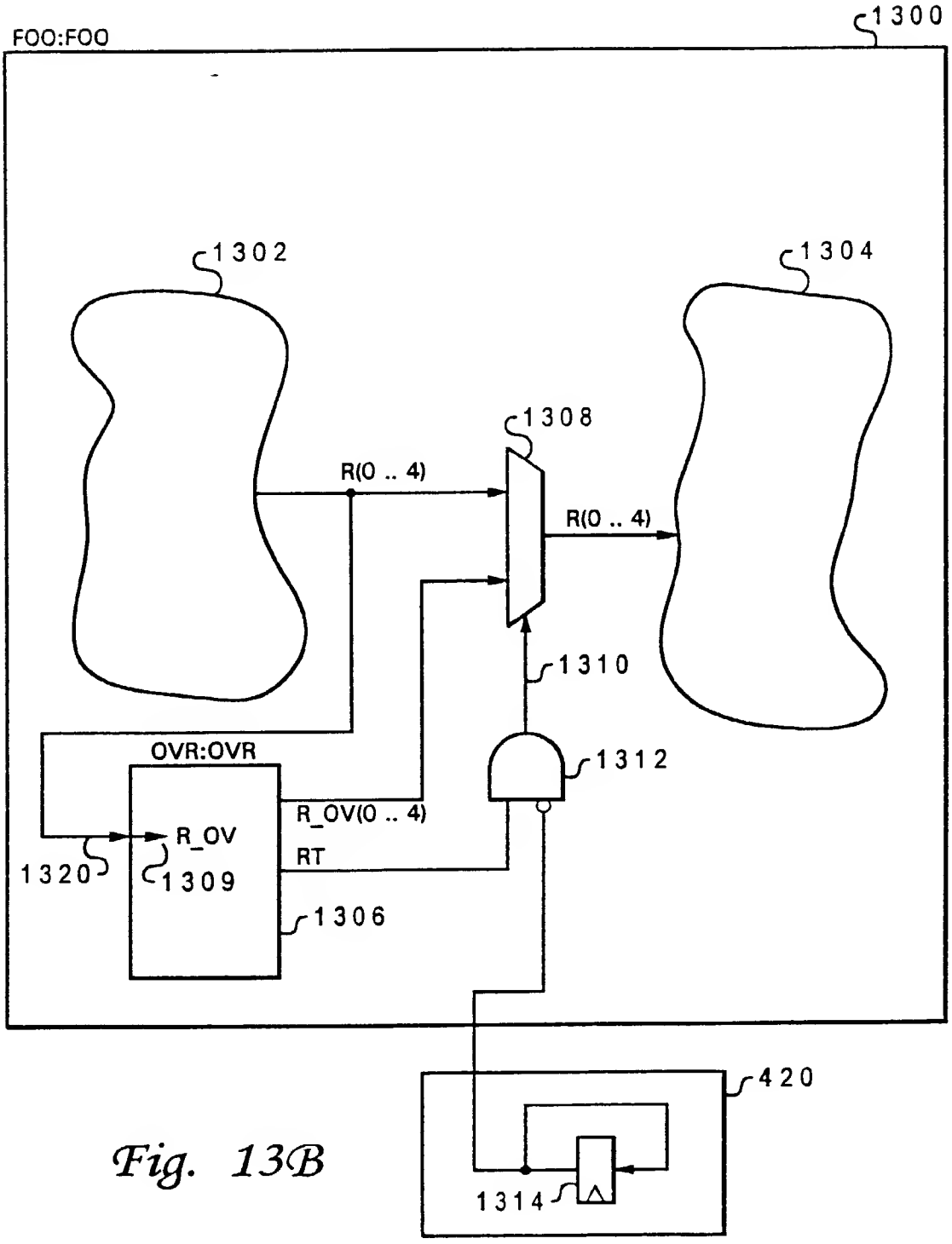

FOO:FOO
1300
1302
1304
1308
R(0 .. 4)
R(0 .. 4)
1310
1312
OVR:OVR
R_OV(0 .. 4)
R_OV
RT
1320
1309
1306
420
1314

Fig. 13B

```
ENTITY OVR IS                                          1364
    PORT(   R_IN        :     IN std_ulogic_vector(0 .. 4);
                .
                .
                .
            ... other ports as required ...
                .
                .                                      1362
                .
            R_OV        :     OUT std_ulogic_vector(0 .. 4);
            RT          :     OUT std_ulogic
        );                                    1363

--!! BEGIN
--!! Design Entity: FOO;

--!! Inputs (0 to 4)                          1360
--!! R_IN => {R(0 .. 4)};
--!! :
... other ports as needed ...
--!! :
--!! End Inputs
                                       1361
--!! Outputs
--!! <R_OVRRIDE> : R_OV(0 .. 4) => R(0 .. 4) [RT];
--!! End Outputs

--!! End

ARCHITECTURE example of OVR IS

BEGIN

    ... HDL code for entity body section ...

END;
```

1356 — 1351 — 1340 — 1358

Fig. 13C

```
ENTITY FOO IS

    PORT(    :
             :
             :
         );

ARCHITECTURE example of FOO IS

BEGIN

    .
    .
    .
    .
    .
    R <= ......
    .
    .
    .
    .
    .
```

1380 {

```
    -!! R_IN <= {R};                                          (1381)
    -!!                                                       (1382)
    -!!
    -!! R_OV(0 to 4) <= ......;                               (1383)
    -!! RT <= ......;
    -!! [override, R_OVRRIDE, R(0 .. 4), RT] <= R_OV(0 to 4);
                                                              (1384)
```

Fig. 13D

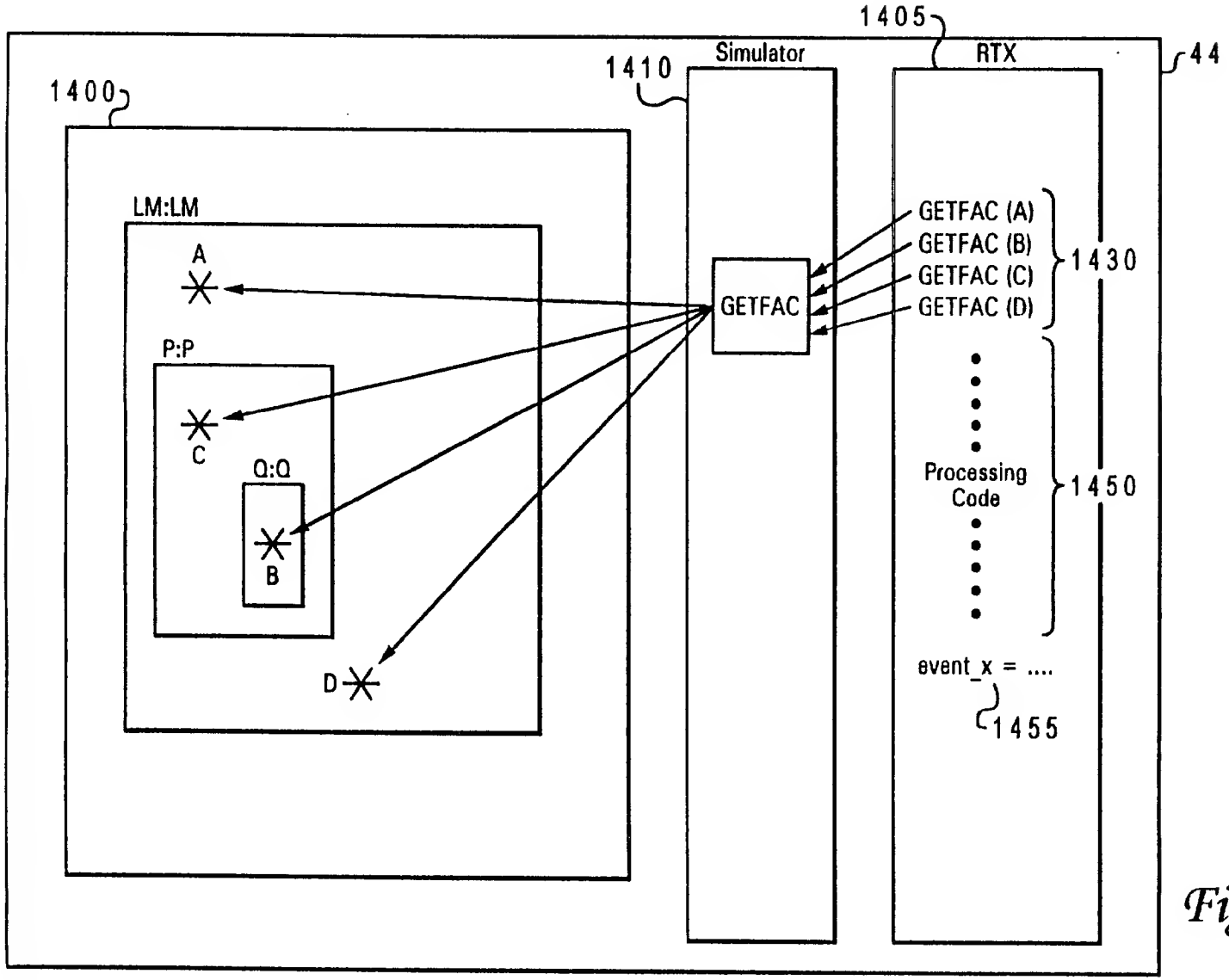

1405
Simulator
RTX
44
1410
1400
LM:LM
A
P:P
C
Q:Q
B
D
GETFAC
GETFAC (A)
GETFAC (B)
GETFAC (C)
GETFAC (D)
1430
Processing
Code
1450
event_x = ...
1455

Fig. 14A

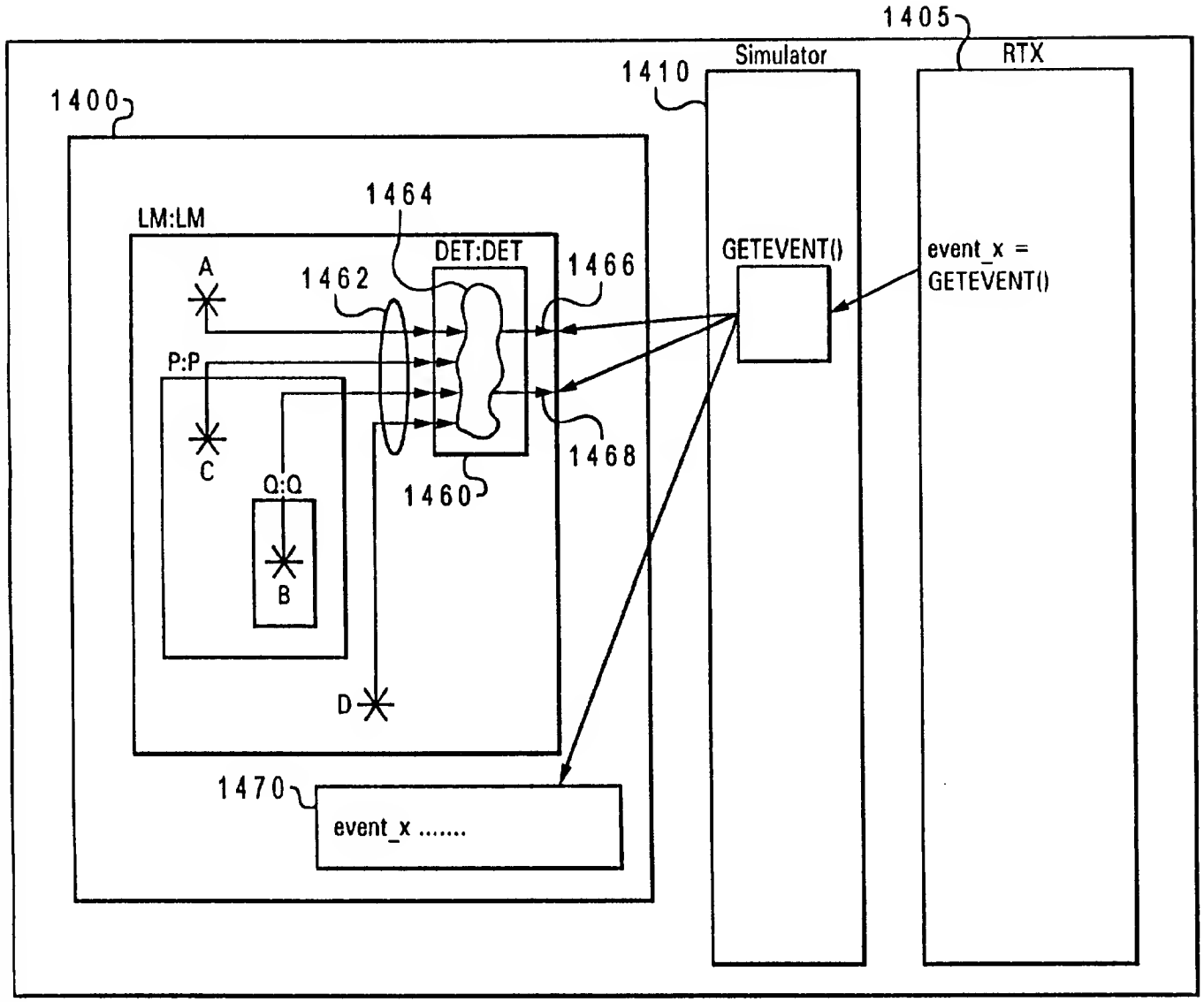


Fig. 14B

```
ENTITY DET IS

    PORT(   A           :   IN std_ulogic;
            B           :   IN std_ulogic_vector(0 to 5);
            C           :   IN std_ulogic;
            D           :   IN std_ulogic;
                        :
                        :
            event_x     :   OUT std_ulogic_vector(0 to 2);
            x_here      :   OUT std_ulogic;
        );

--!! BEGIN
--!! Design Entity: LM;

--!! Inputs
--!! A     =>    A;
--!! B     =>    P.Q.B;
--!! C     =>    P.C;
--!! D     =>    D;
--!! End Inputs

--!! Detections
--!! <event_x>:event_x(0 to 2) [x_here];
--!! End Detections

--!! End;

ARCHITECTURE example of DET IS

BEGIN

    ... HDL code ...

END;
```

1491 1493 1495 1494 1492 1480

Fig. 14C

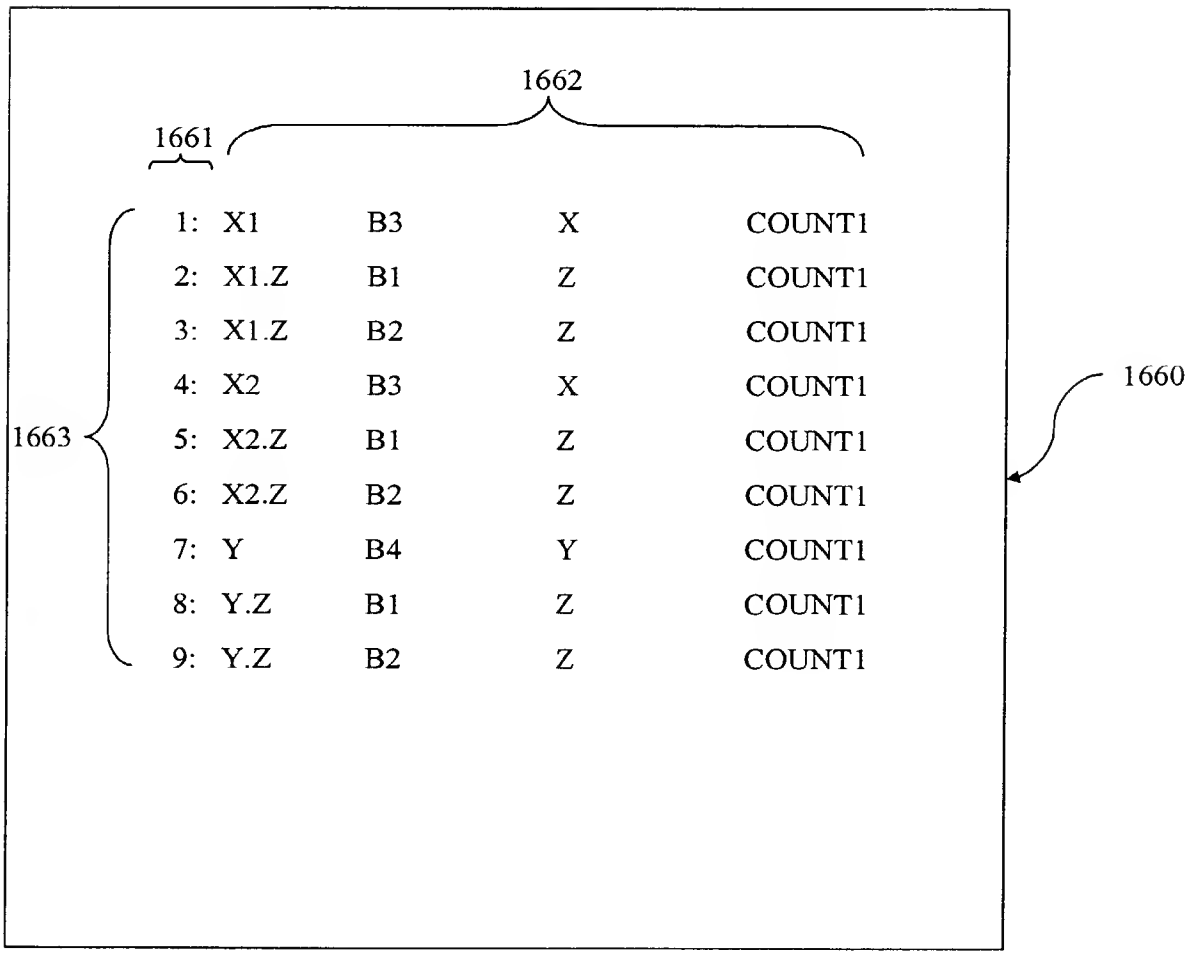

1662
1661
1: X1 B3 X COUNT1
2: X1.Z B1 Z COUNT1
3: X1.Z B2 Z COUNT1
4: X2 B3 X COUNT1
5: X2.Z B1 Z COUNT1
6: X2.Z B2 Z COUNT1
7: Y B4 Y COUNT1
8: Y.Z B1 Z COUNT1
9: Y.Z B2 Z COUNT1
1663
1660

FIG. 15

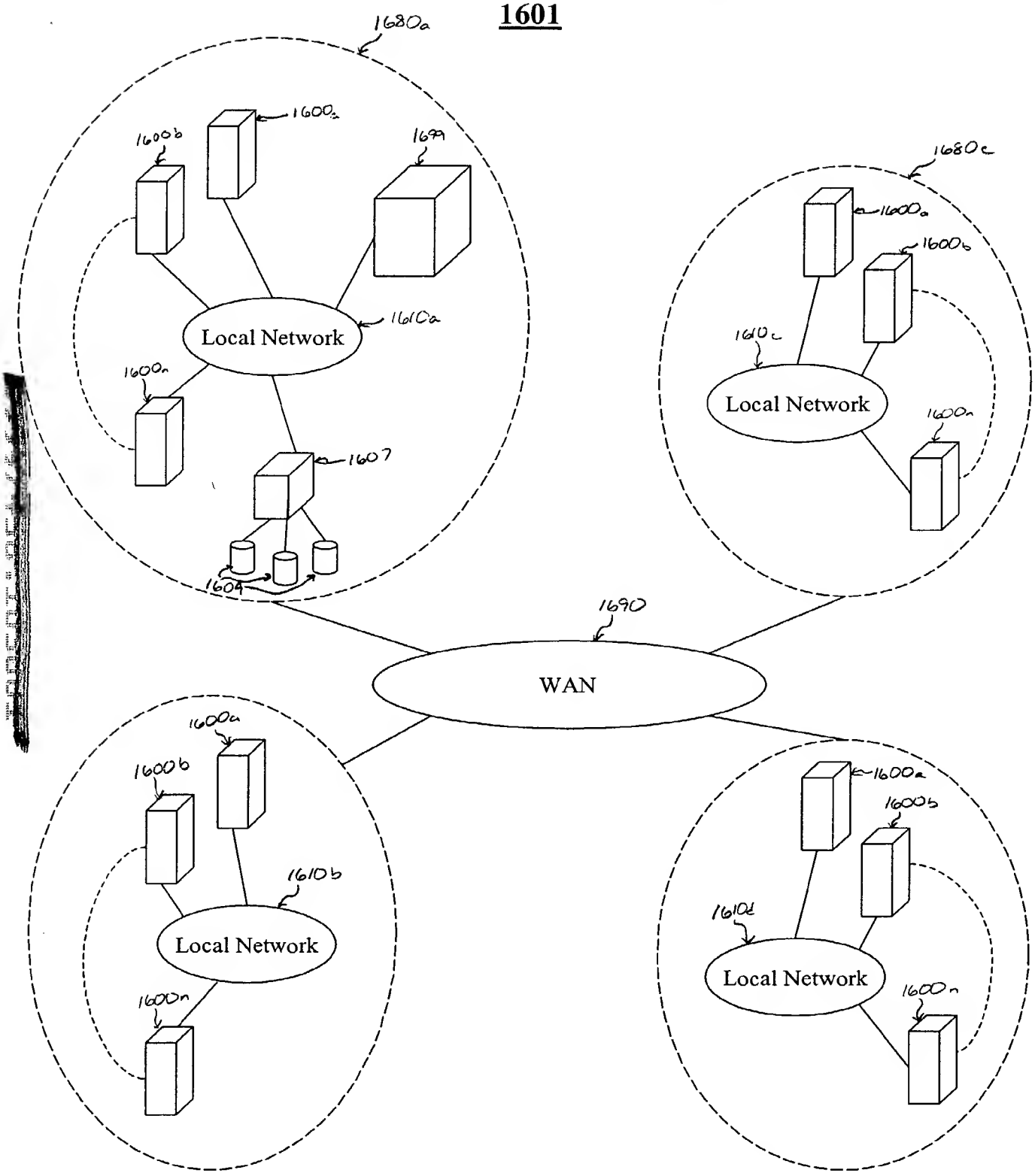

1601
1680a
1600a
1600b
1699
Local Network
1610a
1600n
1607
1609
1680c
1600a
1600b
1610c
Local Network
1600n
1690
WAN
1600a
1600b
1610b
Local Network
1600n
1600a
1600b
1610d
Local Network
1600n

FIG. 16B

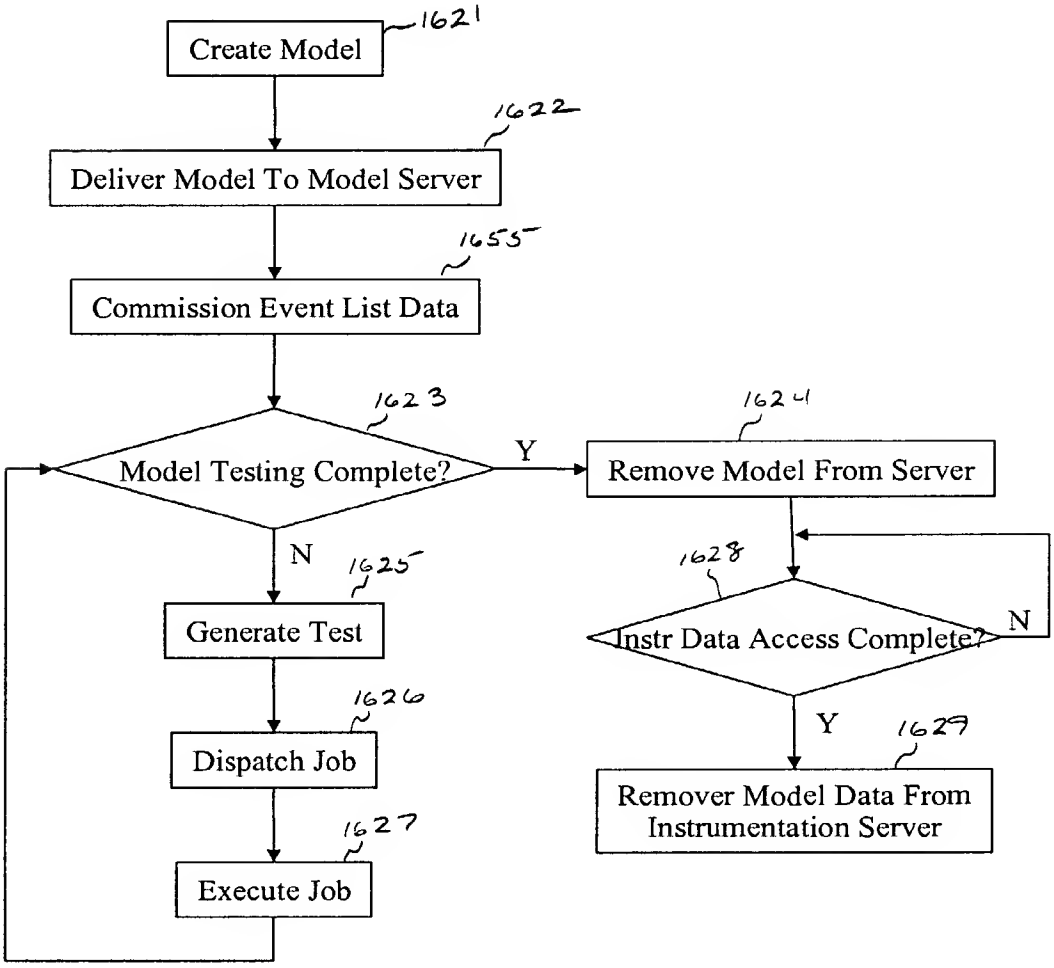

Create Model
1621
Deliver Model To Model Server
1622
Commission Event List Data
1623
Model Testing Complete?
Y
N
Generate Test
1625
Dispatch Job
1626
Execute Job
1627
Remove Model From Server
1624
Instr Data Access Complete?
1628
N
Y
Remover Model Data From Instrumentation Server
1629

FIG. 16C

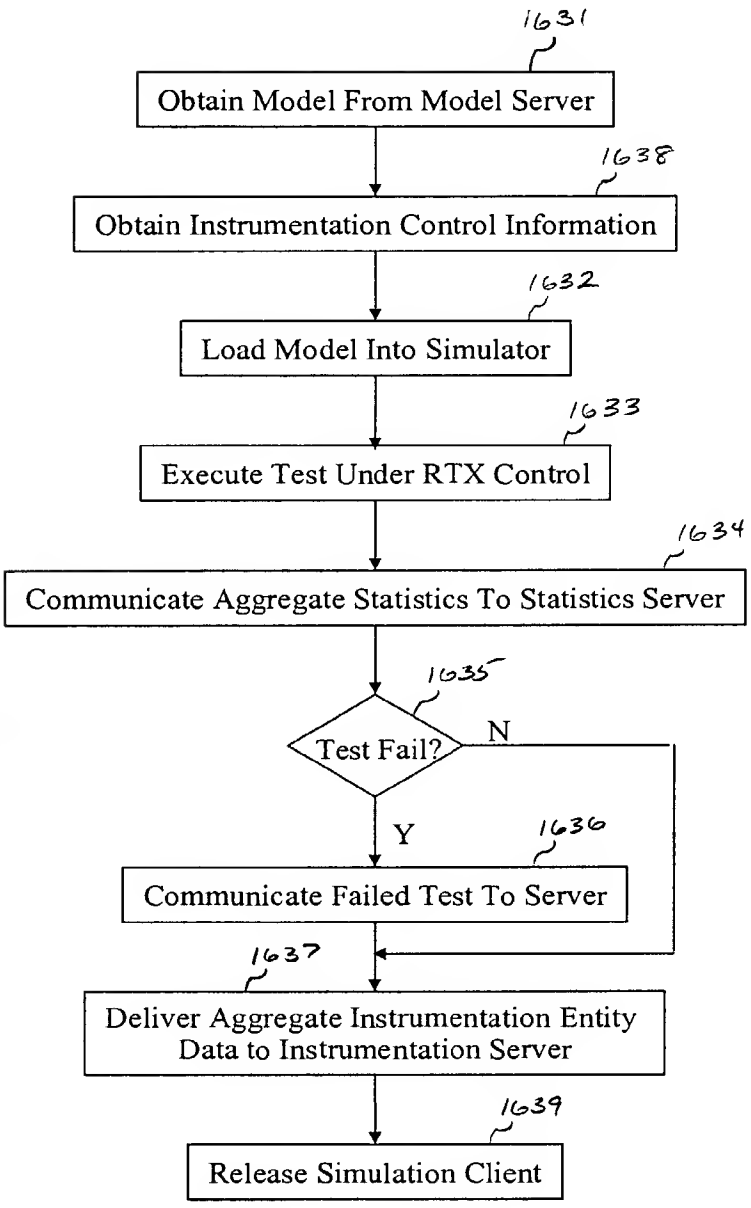

1631
Obtain Model From Model Server
1638
Obtain Instrumentation Control Information
1632
Load Model Into Simulator
1633
Execute Test Under RTX Control
1634
Communicate Aggregate Statistics To Statistics Server
1635
Test Fail?
N
Y
1636
Communicate Failed Test To Server
1637
Deliver Aggregate Instrumentation Entity Data to Instrumentation Server
1639
Release Simulation Client

FIG. 16D

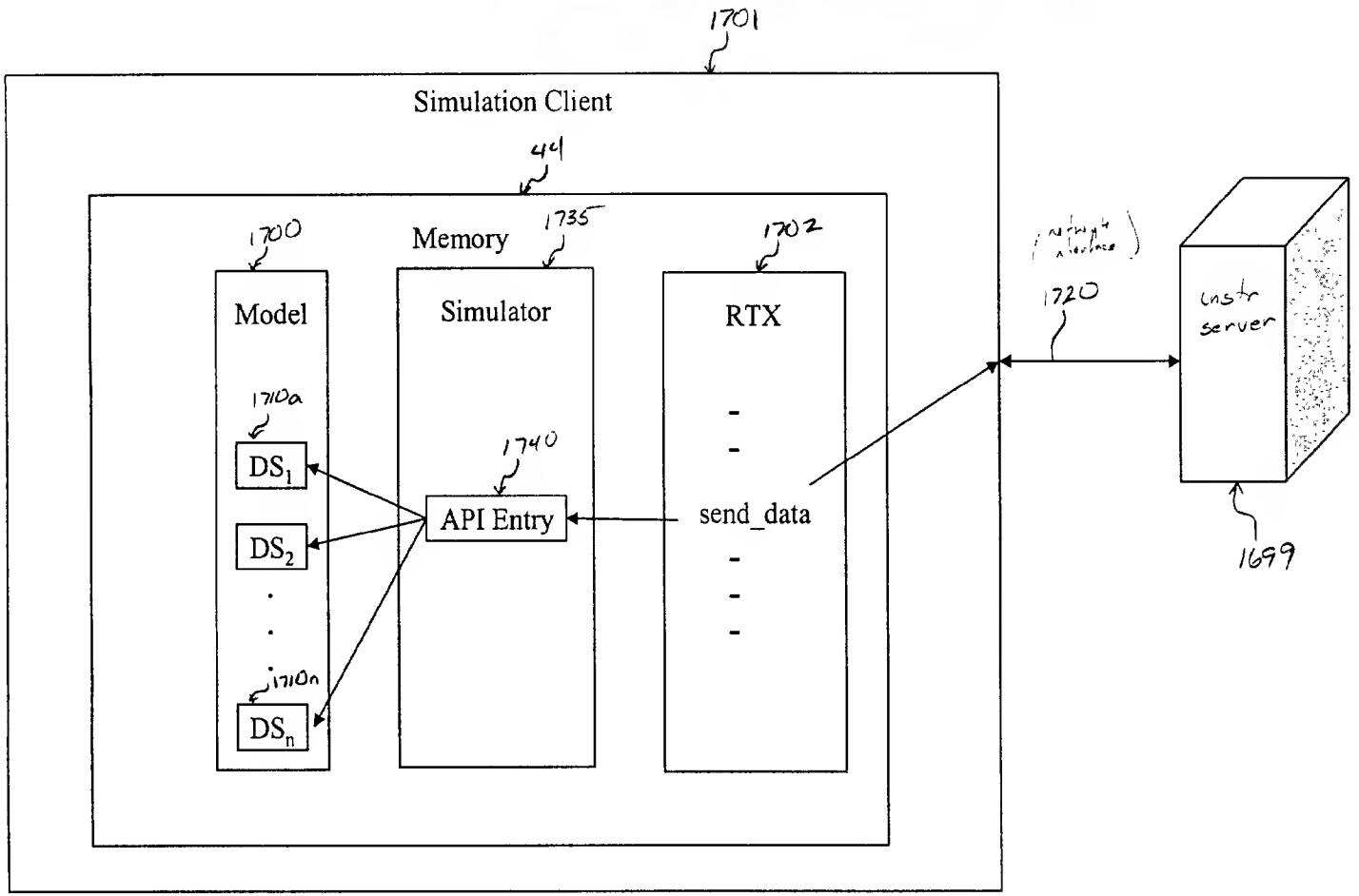


FIG. 17A

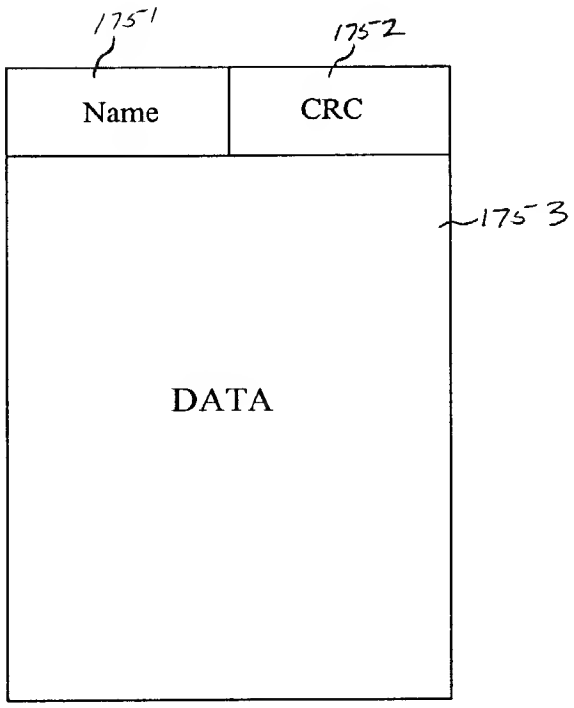


FIG. 17B

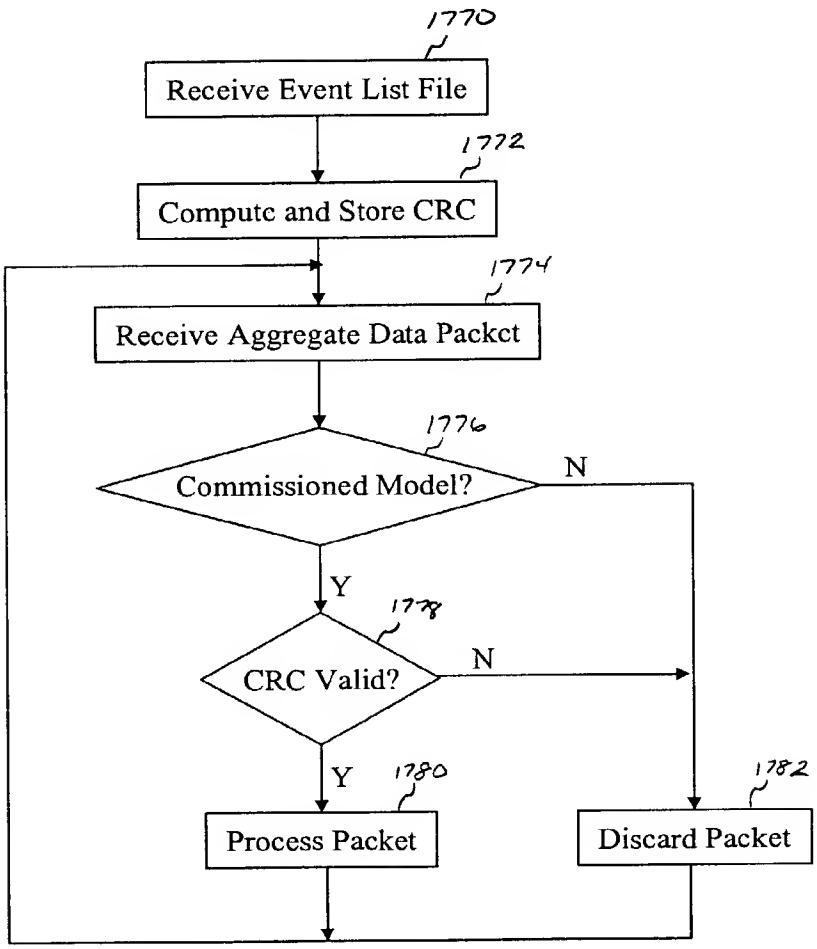

1770
Receive Event List File
1772
Compute and Store CRC
1774
Receive Aggregate Data Packet
1776
Commissioned Model?
N
Y
1778
CRC Valid?
N
Y
1780
Process Packet
1782
Discard Packet

FIG. 17C

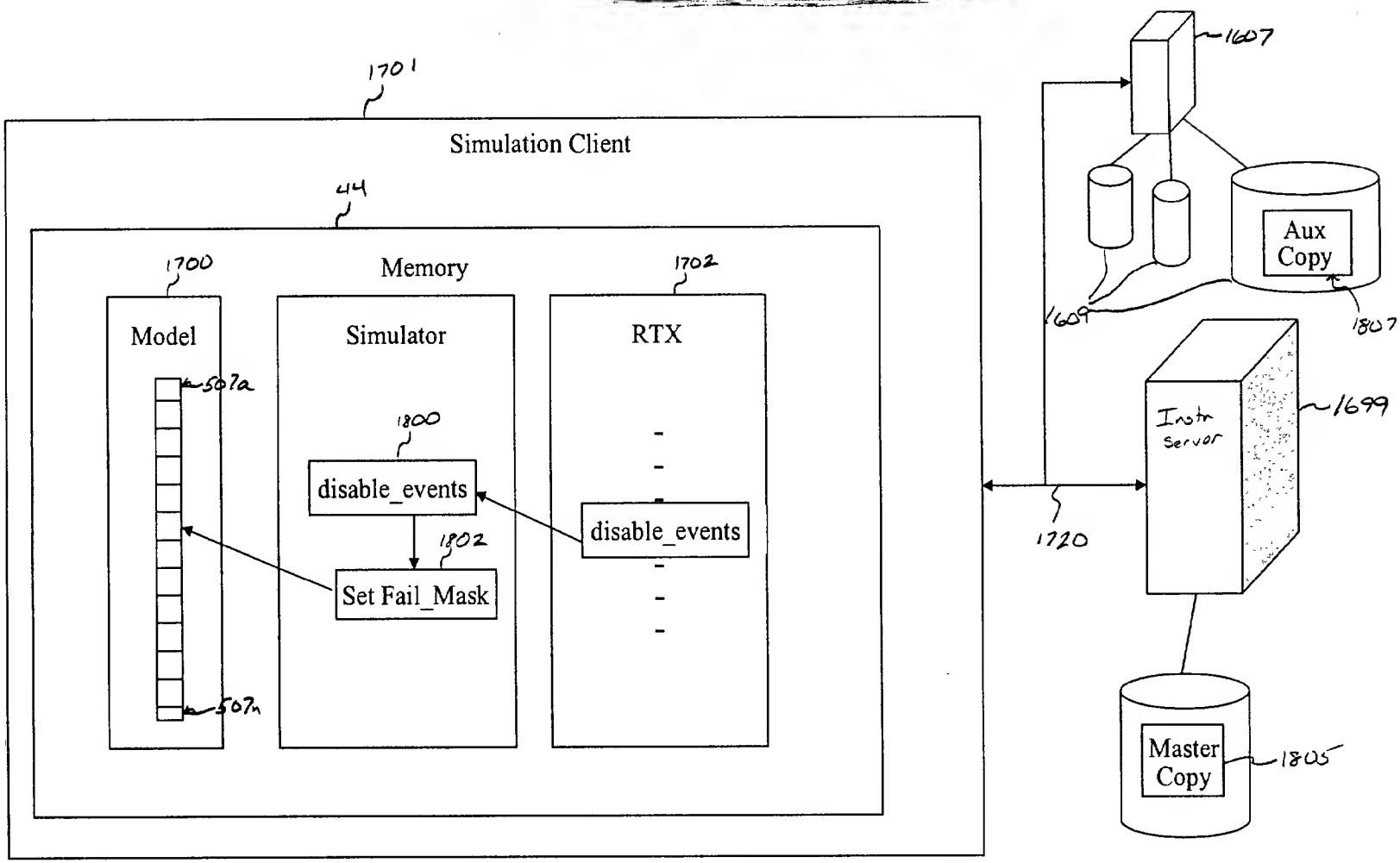


FIG. 18A

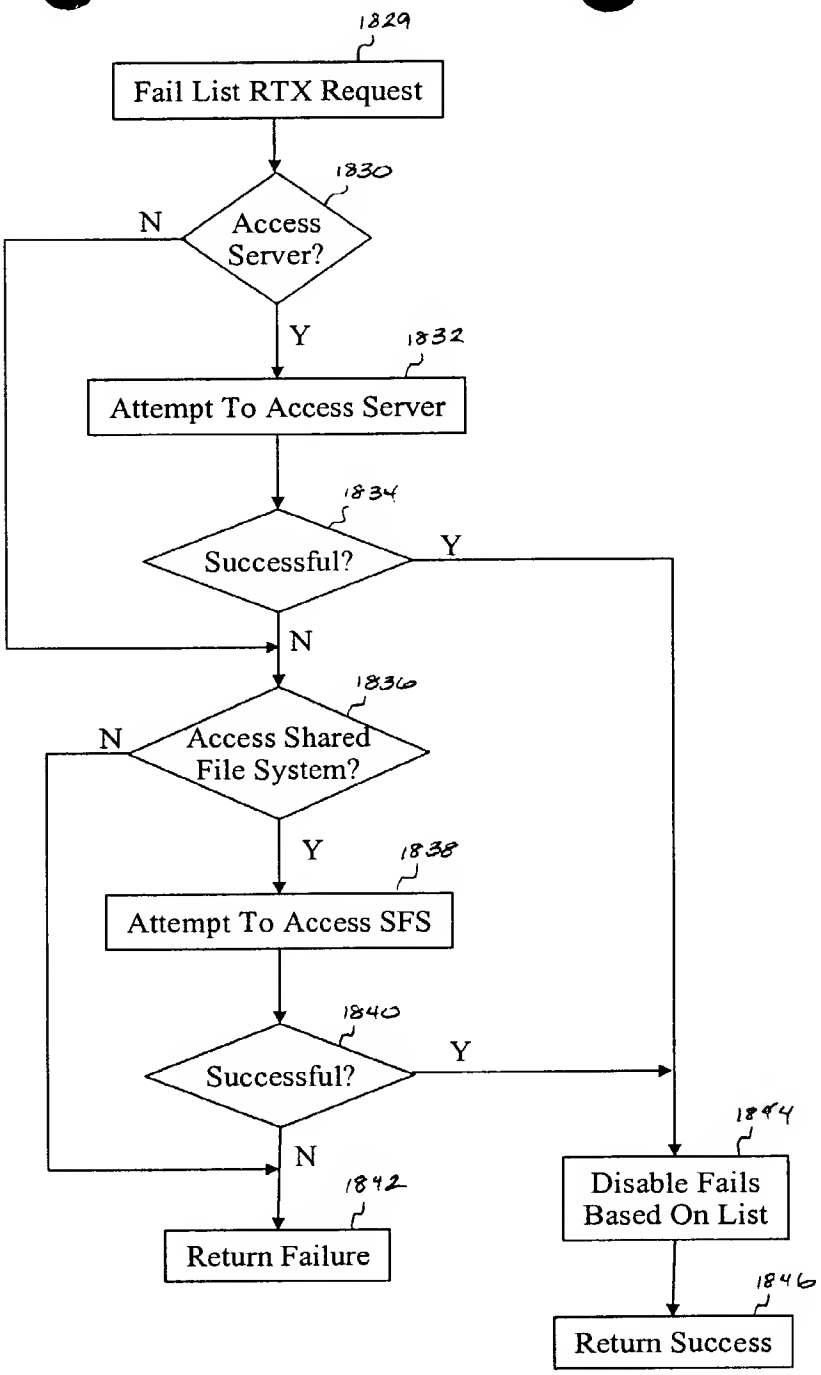


FIG. 18B

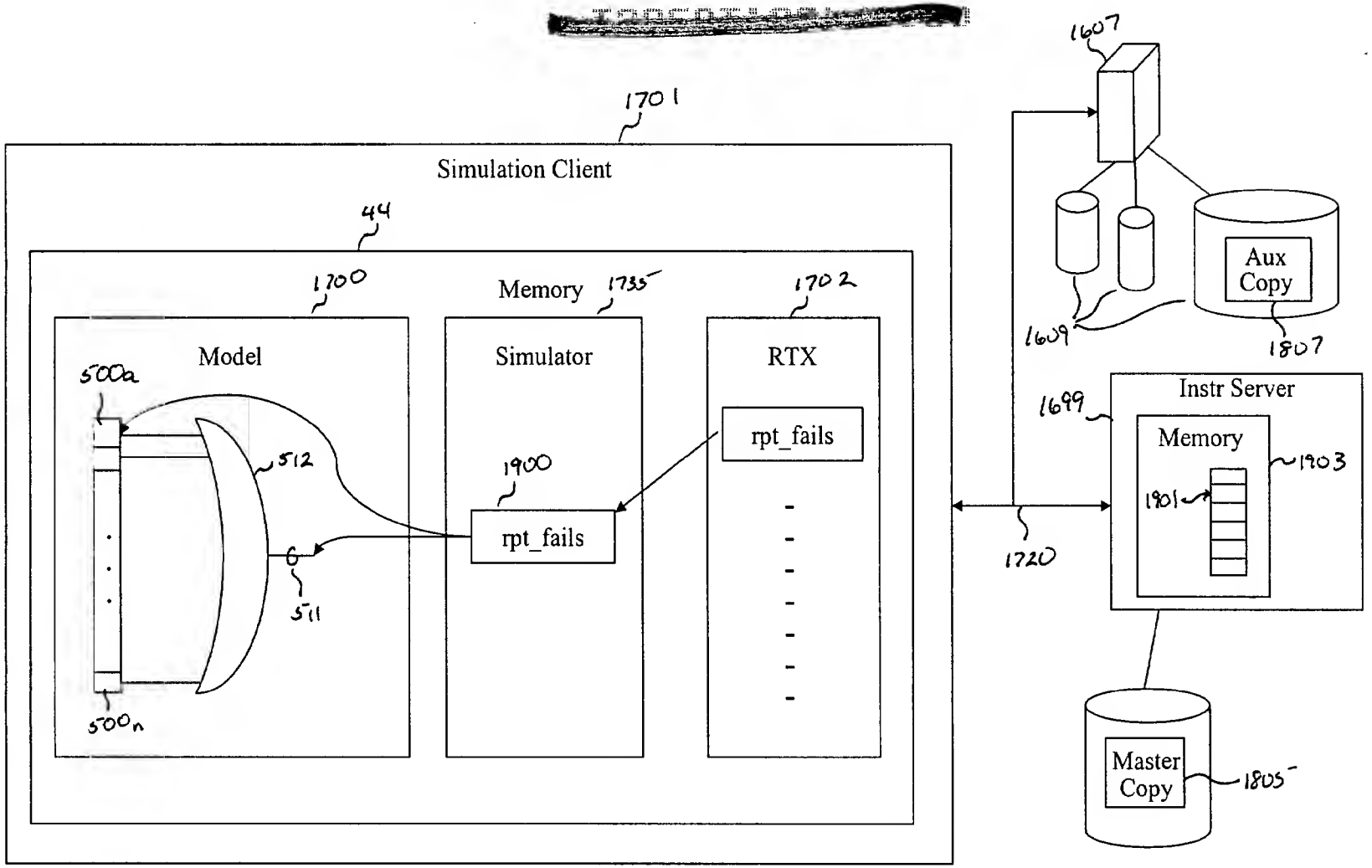


FIG. 19A

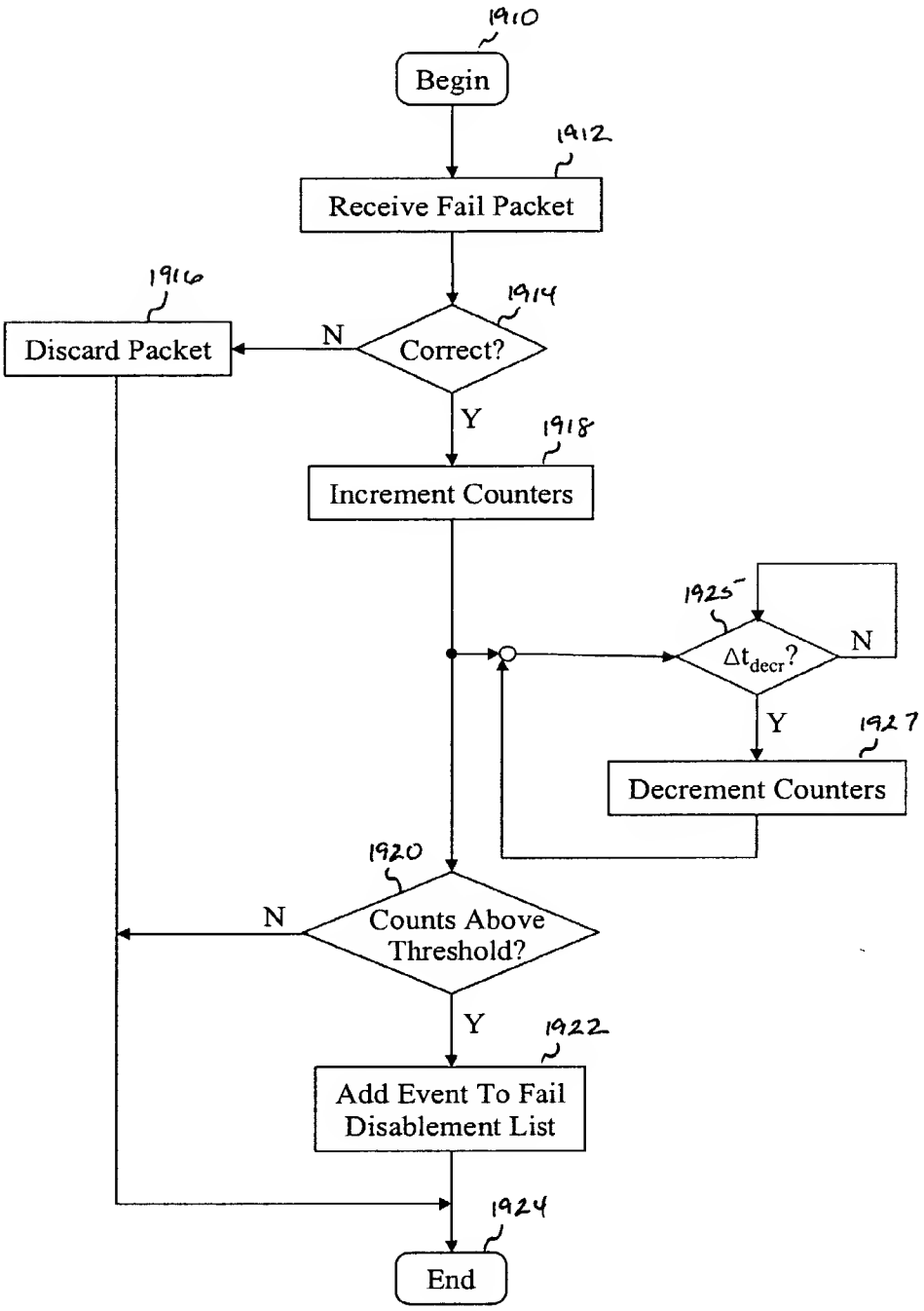

1910
Begin
1912
Receive Fail Packet
1916
Discard Packet
N
1914
Correct?
Y
1918
Increment Counters
1925
Δt_{decr}?
N
Y
1927
Decrement Counters
1920
N
Counts Above Threshold?
Y
1922
Add Event To Fail Disablement List
1924
End

FIG. 19B

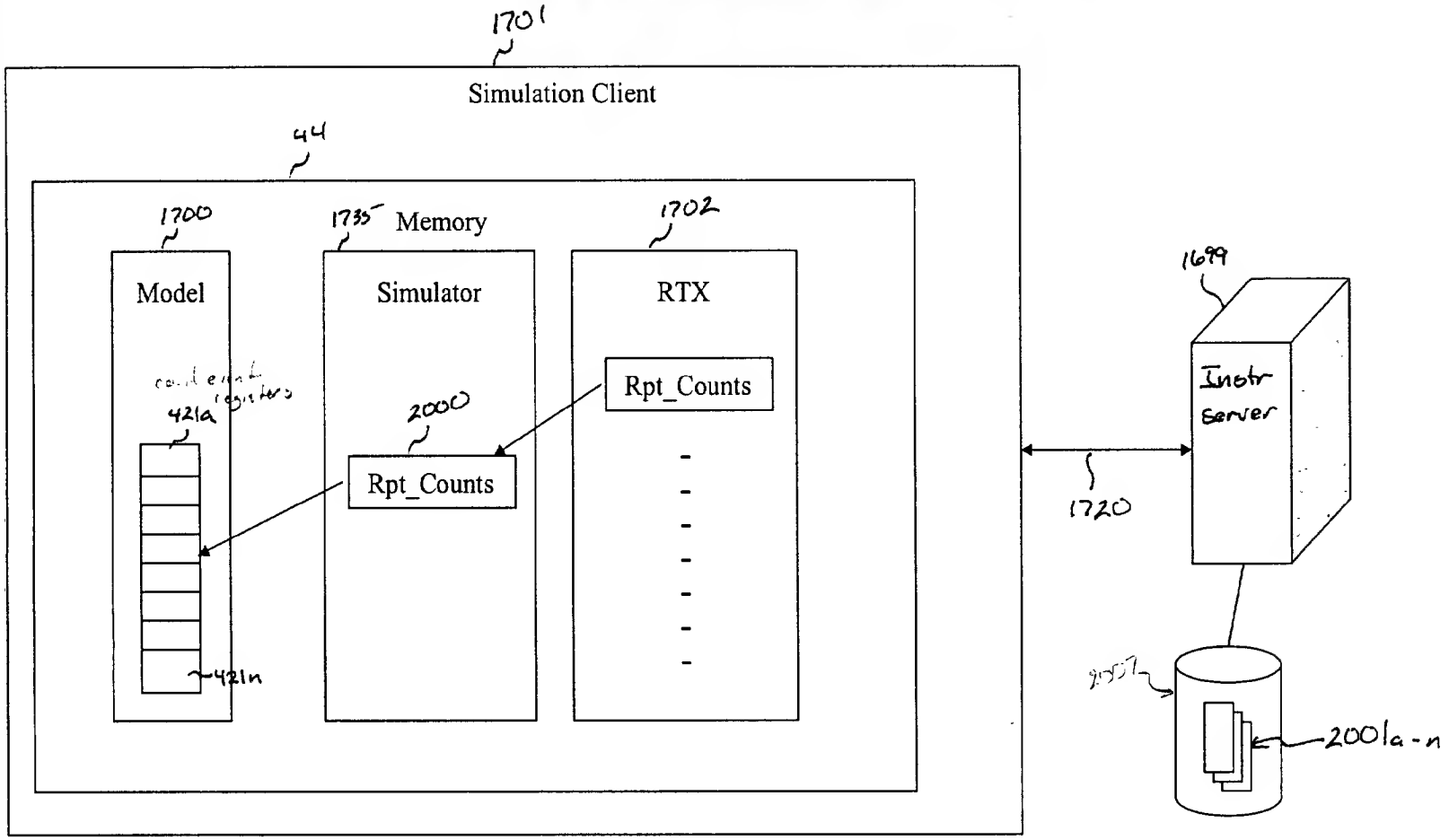


FIG. 20A

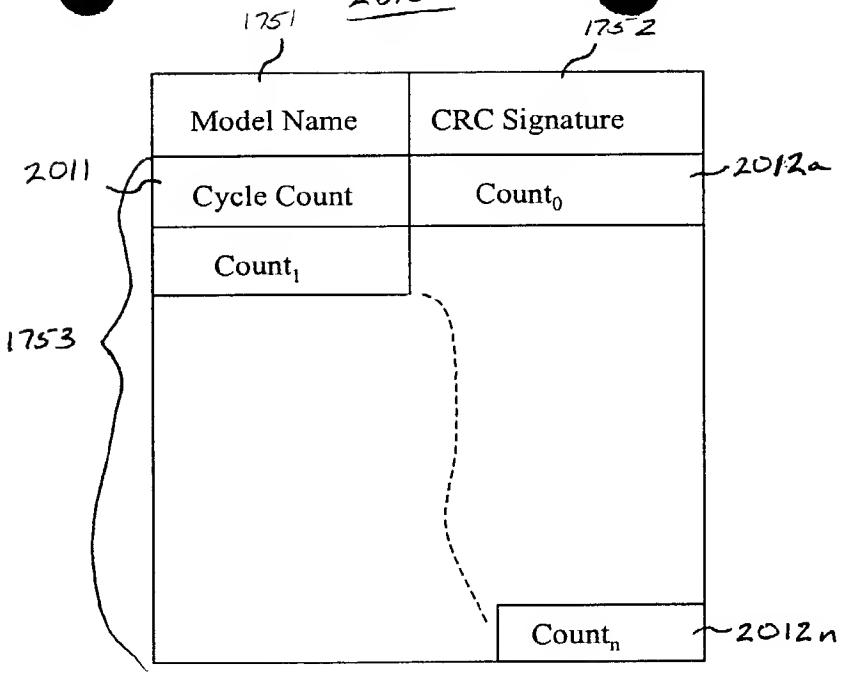


FIG. 20B

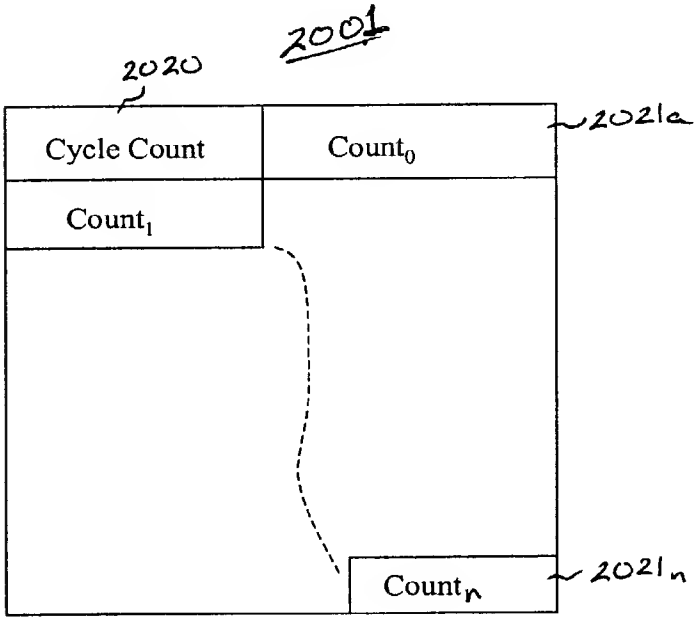


FIG. 20C

FIG. 20D

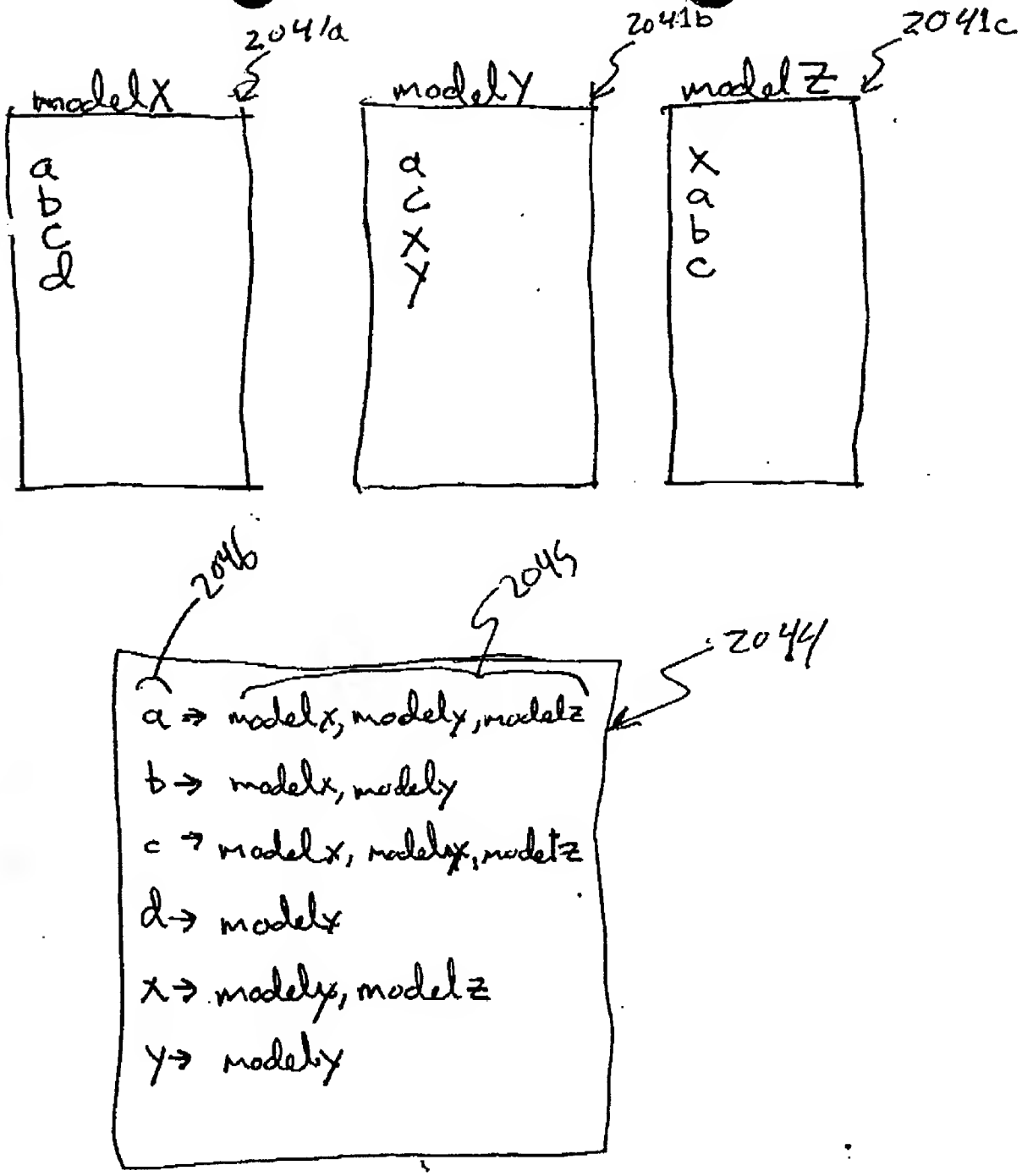


FIG. 20E

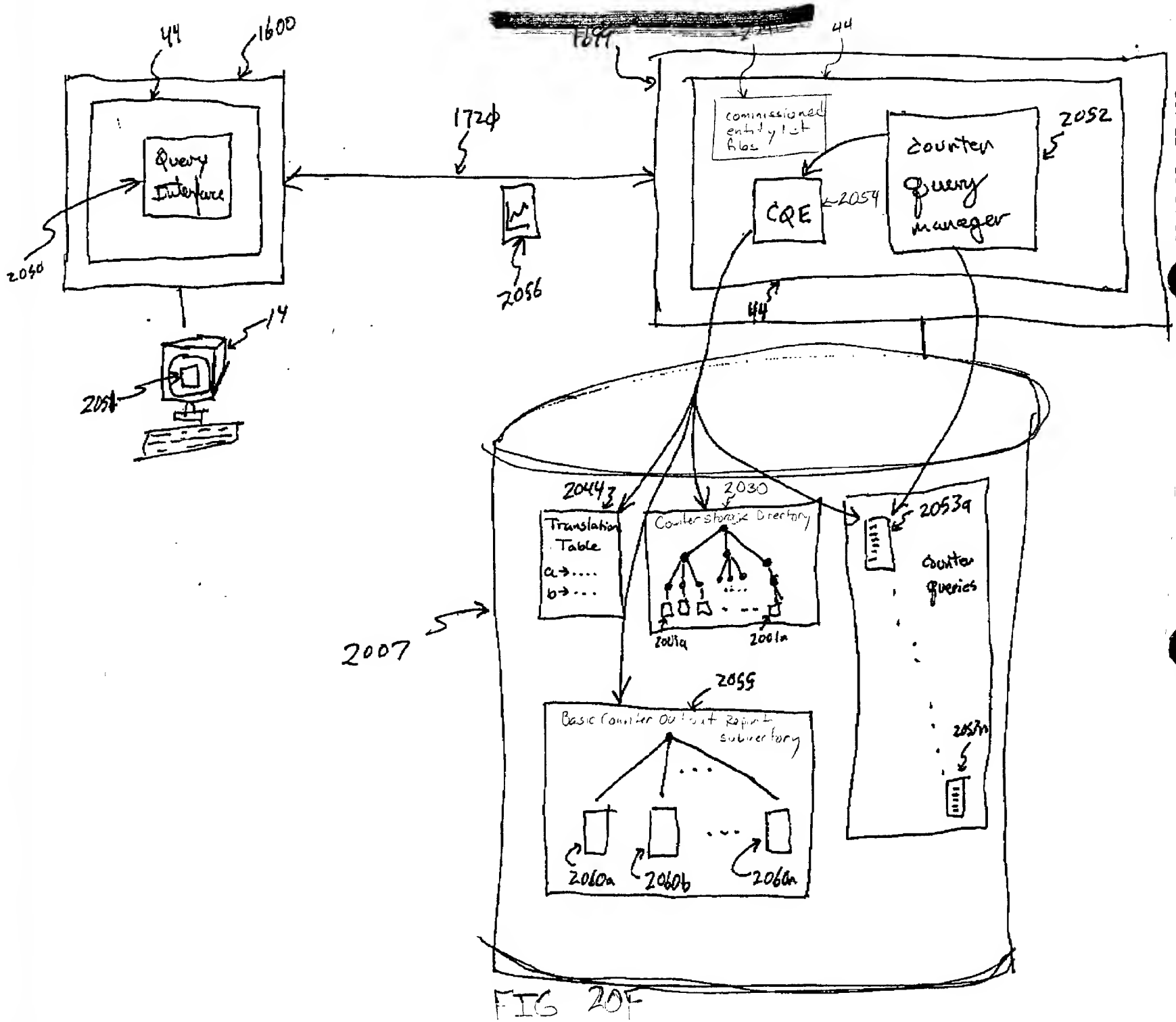


FIG 20F

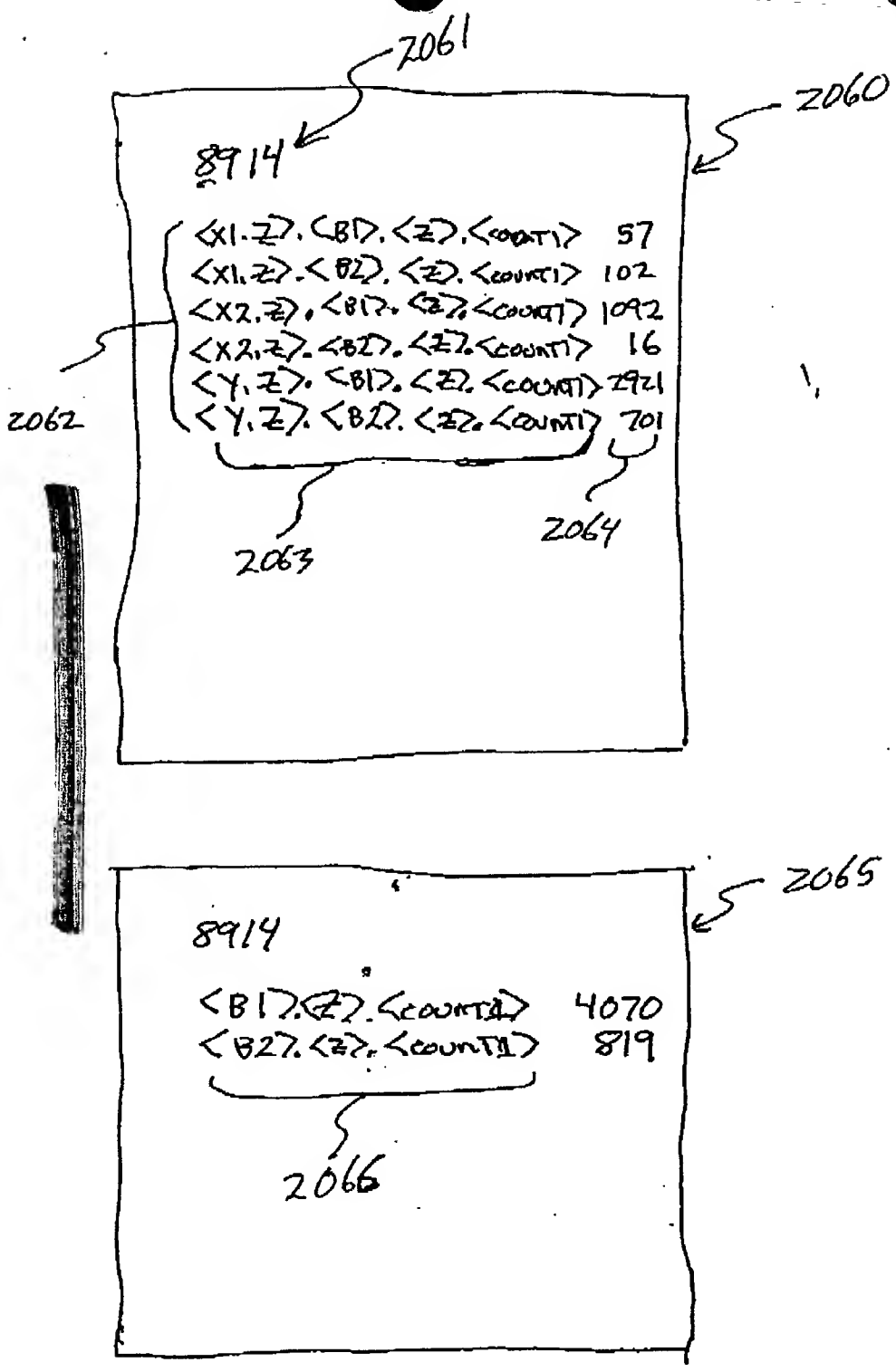

2061
2060
8914
<X1.Z>.<B1>.<Z>.<COUNT1> 57
<X1.Z>.<B2>.<Z>.<COUNT1> 102
<X2.Z>.<B1>.<Z>.<COUNT1> 1092
<X2.Z>.<B2>.<Z>.<COUNT1> 16
<Y.Z>.<B1>.<Z>.<COUNT1> 2921
<Y.Z>.<B2>.<Z>.<COUNT1> 701
2062
2063
2064
2065
8914
<B1>.<Z>.<COUNT1> 4070
<B2>.<Z>.<COUNT1> 819
2066

FIG. 206

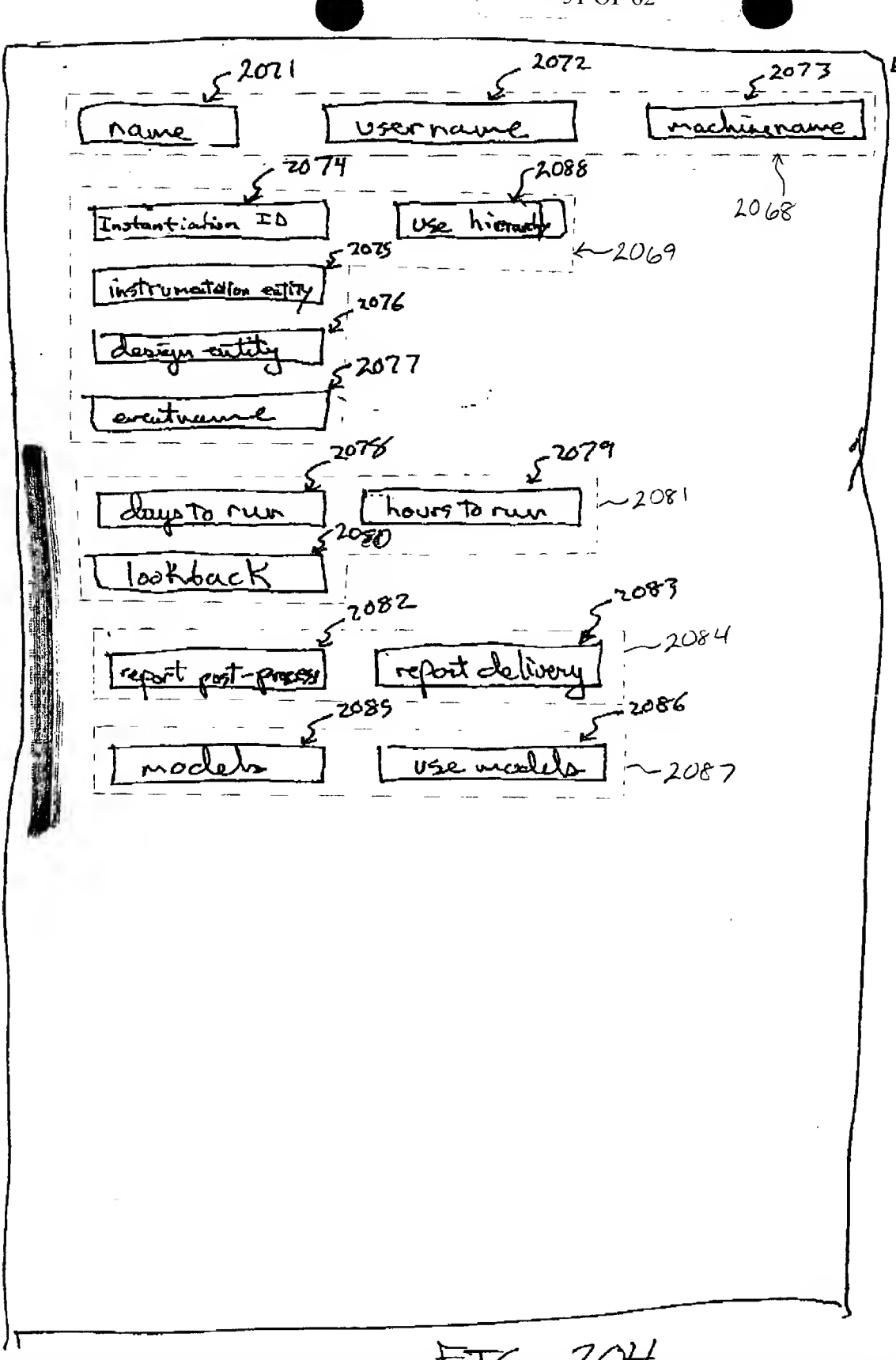

2053
2071
name
2072
username
2073
machinename
2068
2074
Instantiation ID
2088
use hierarchy
2069
2075
instrumentation entity
2076
design entity
2077
executname
2078
days to run
2079
hours to run
2081
2080
lookback
2082
report post-process
2083
report delivery
2084
2085
models
2086
use models
2087

FIG. 20H

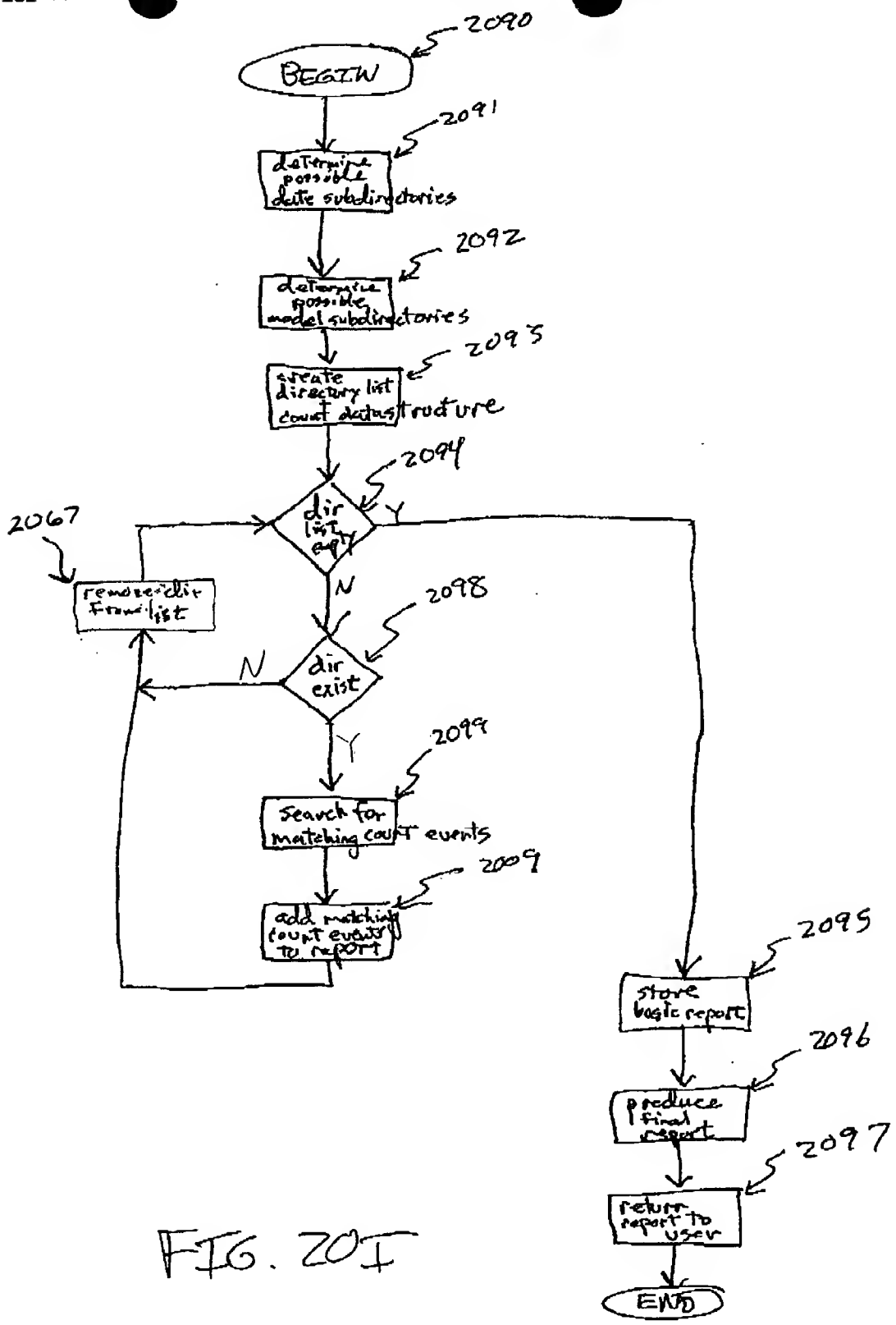


FIG. 20I

2052
counter query manager
CDAE
2080
2060a
2060b
2060n
2055
2105a
2105b
2105n
2106
2107
14
16
17
207

FIG. 21A

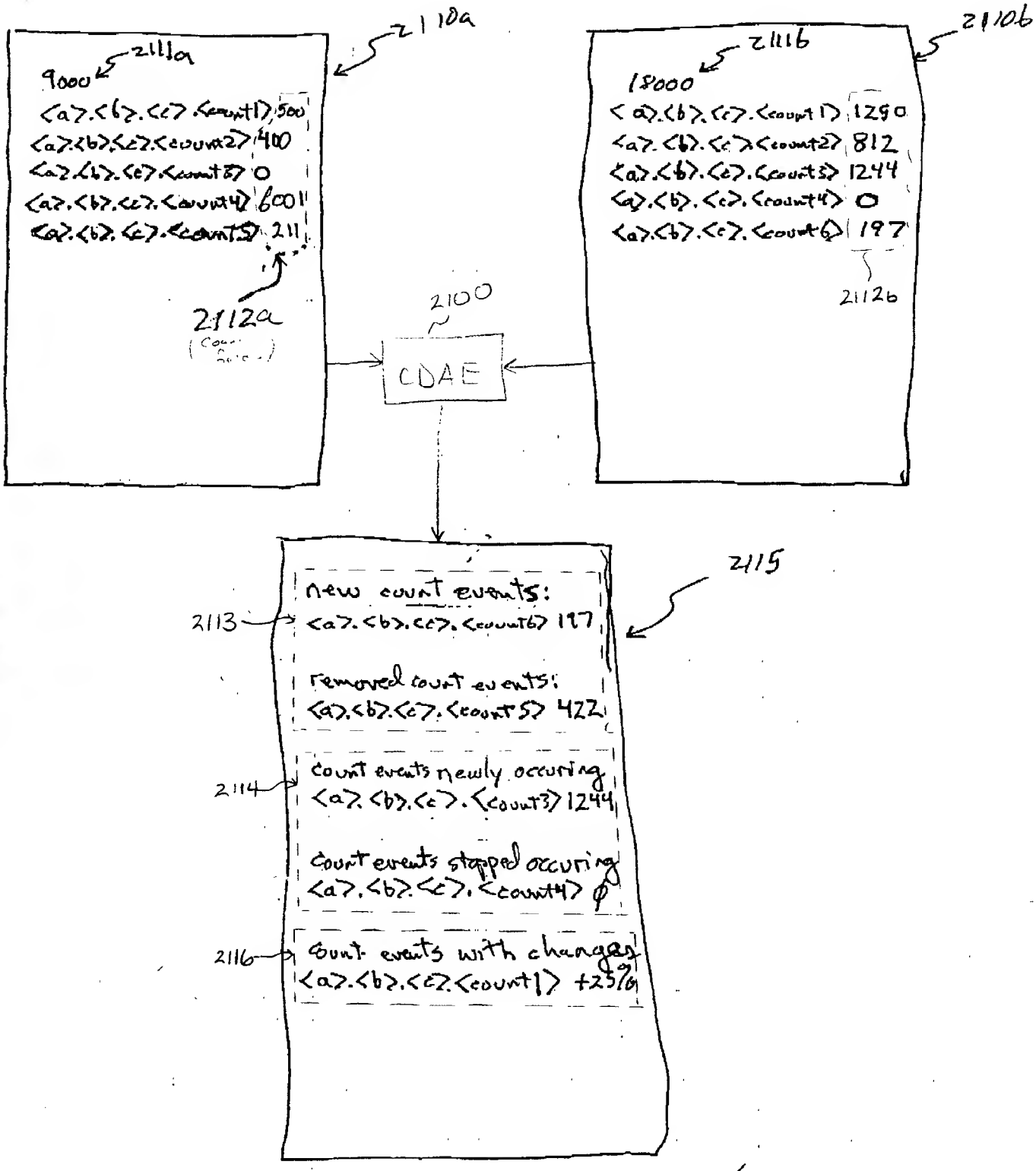


FIG. 21B

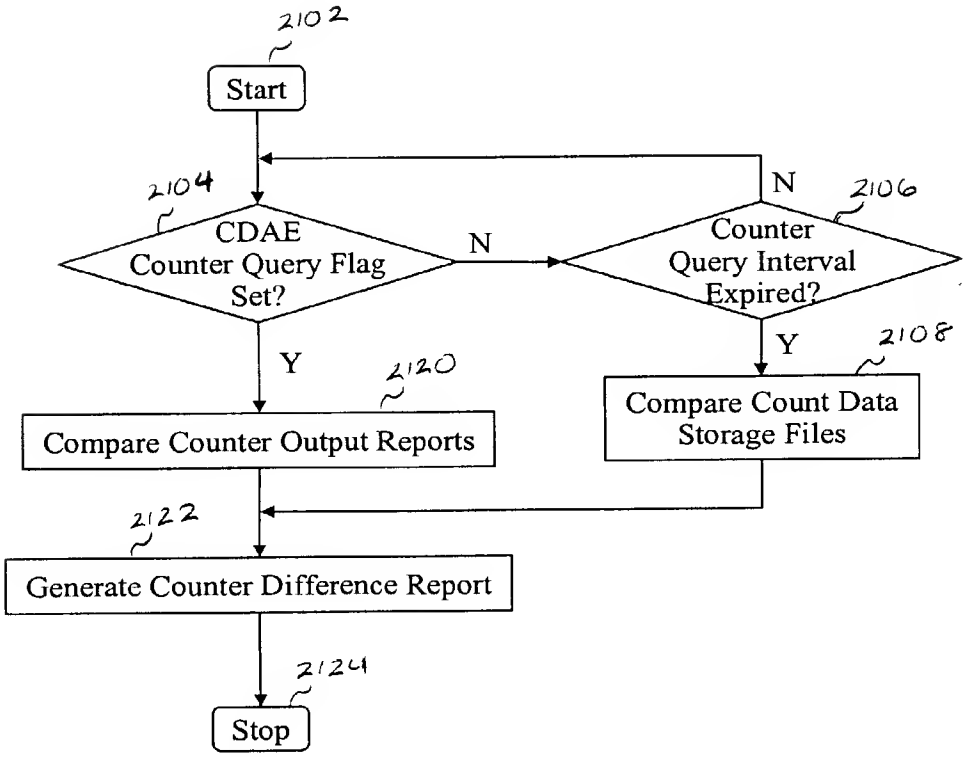

2102
Start
2104
CDAE Counter Query Flag Set?
N
2106
Counter Query Interval Expired?
N
Y
Y
2120
Compare Counter Output Reports
2108
Compare Count Data Storage Files
2122
Generate Counter Difference Report
2124
Stop

FIG. 21C

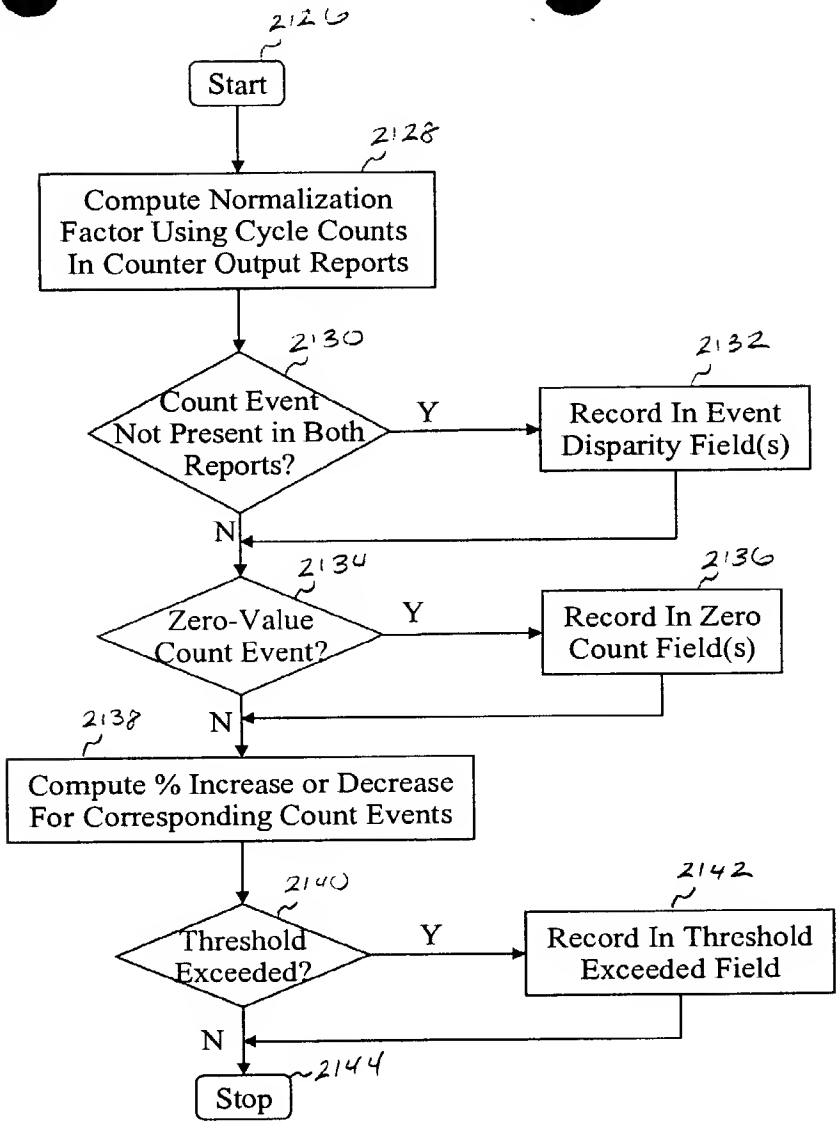

2126
Start
2128
Compute Normalization Factor Using Cycle Counts In Counter Output Reports
2130
Count Event Not Present in Both Reports?
Y
2132
Record In Event Disparity Field(s)
N
2134
Zero-Value Count Event?
Y
2136
Record In Zero Count Field(s)
N
2138
Compute % Increase or Decrease For Corresponding Count Events
2140
Threshold Exceeded?
Y
2142
Record In Threshold Exceeded Field
N
2144
Stop

FIG. 21D

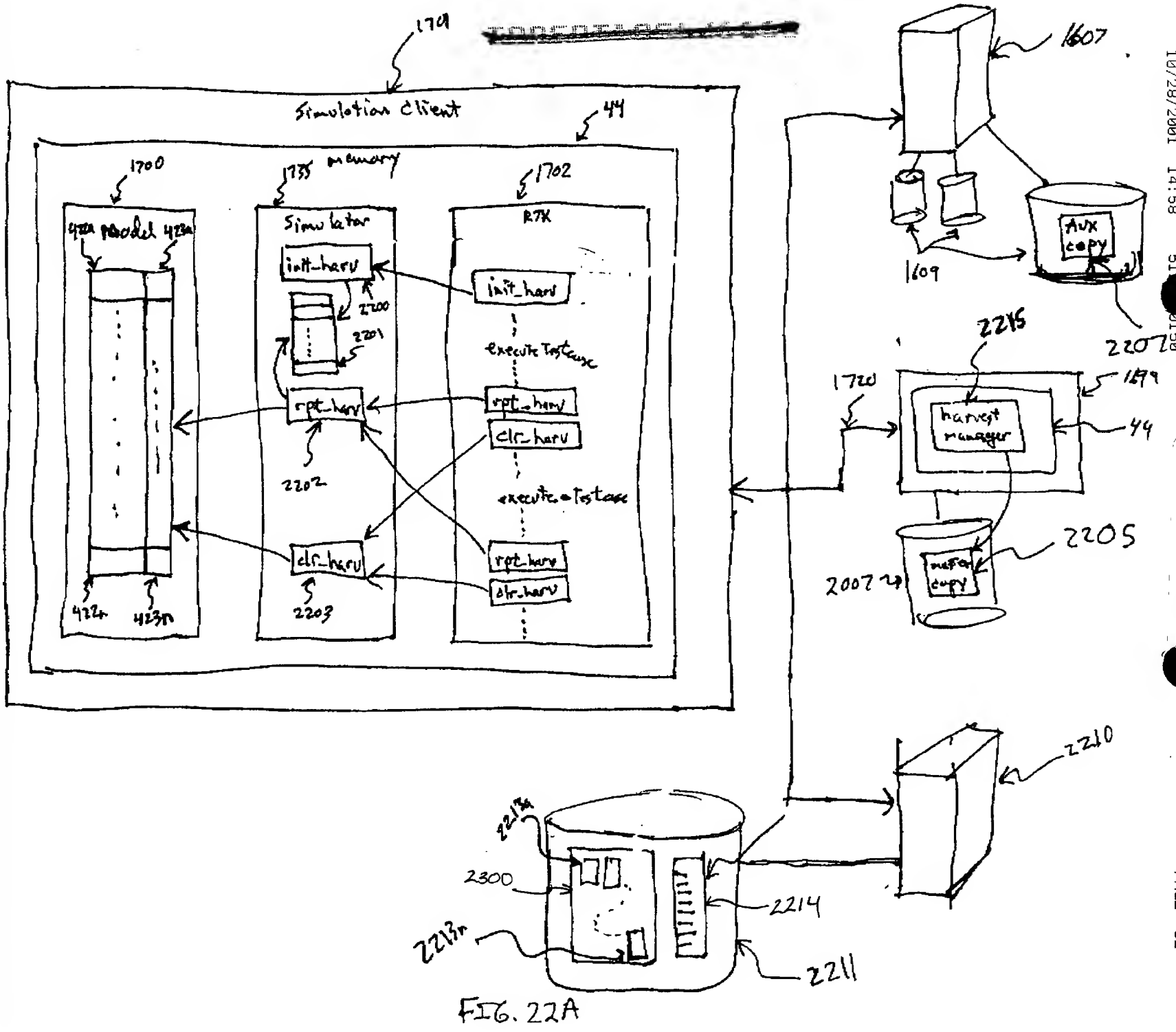


FIG. 22A

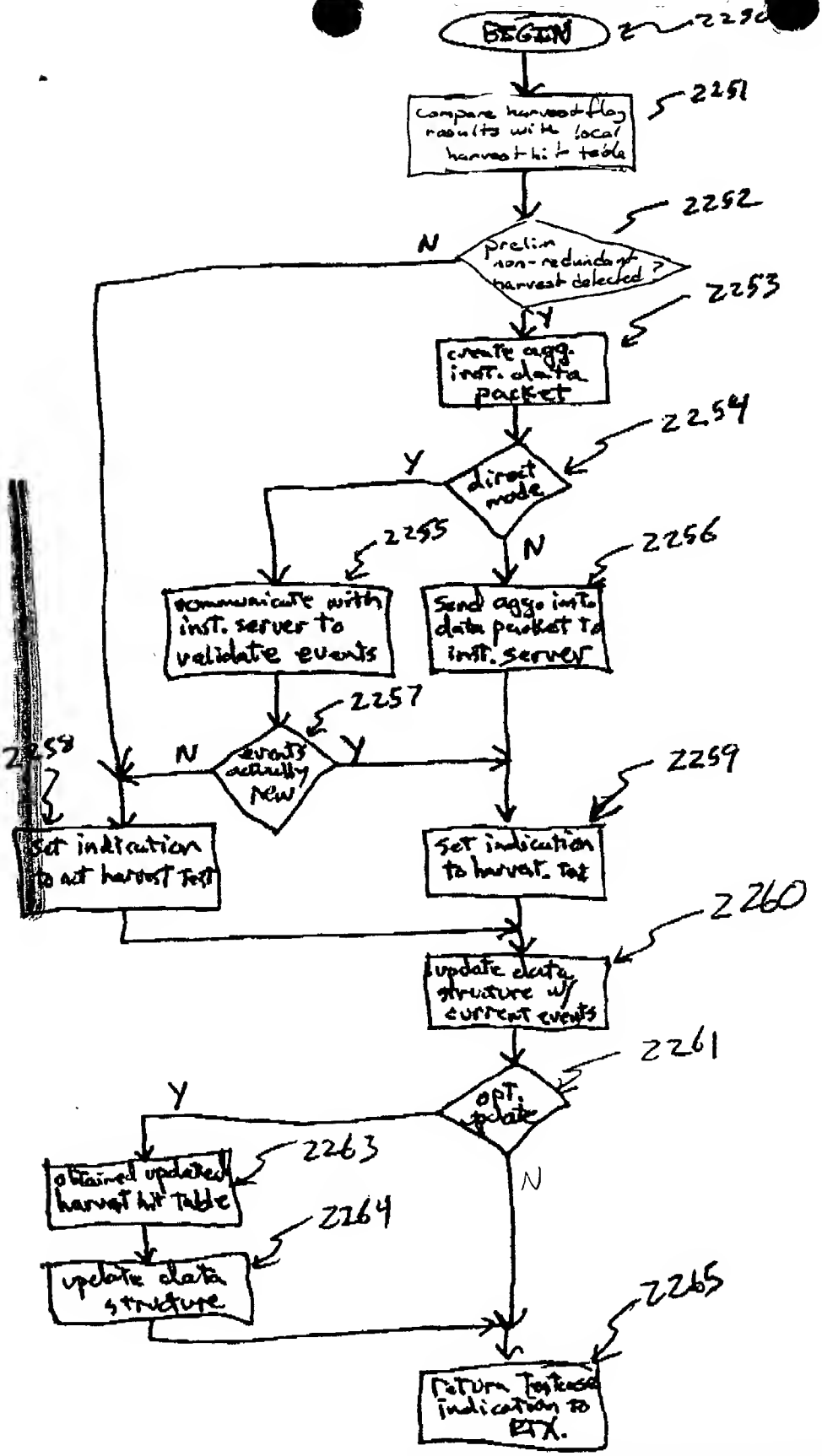


FIG. 22B

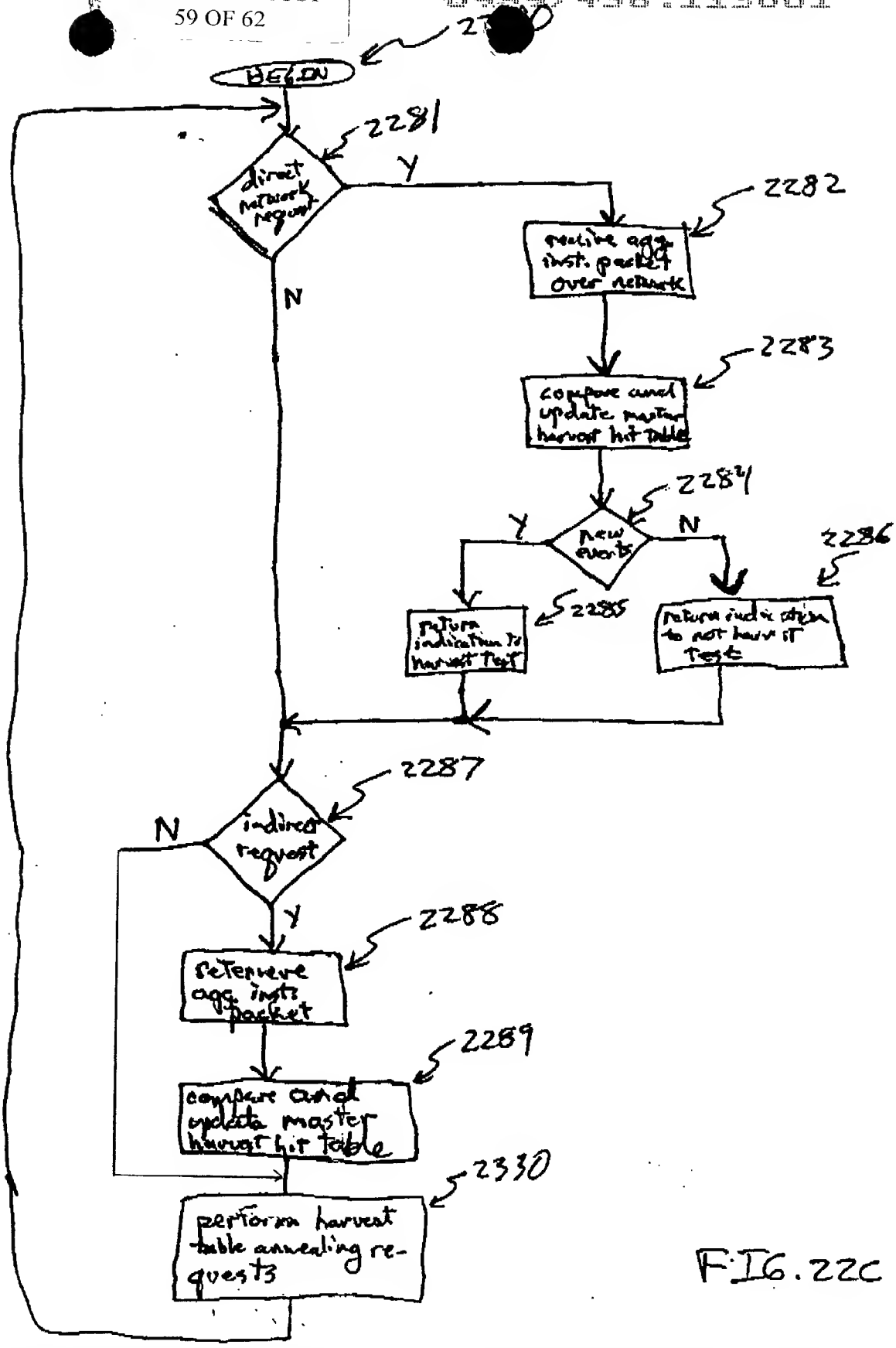

BEGIN
2281
direct network request
Y
N
2282
receive agg. inst. packet over network
2283
compare and update master harvest hit table
2284
new events
Y
N
2285
return indication to harvest Test
2286
return indication to not harvest Test
2287
indirect request
N
Y
2288
retrieve agg. inst. packet
2289
compare and update master harvest hit table
2330
perform harvest table annealing requests

FIG. 22C

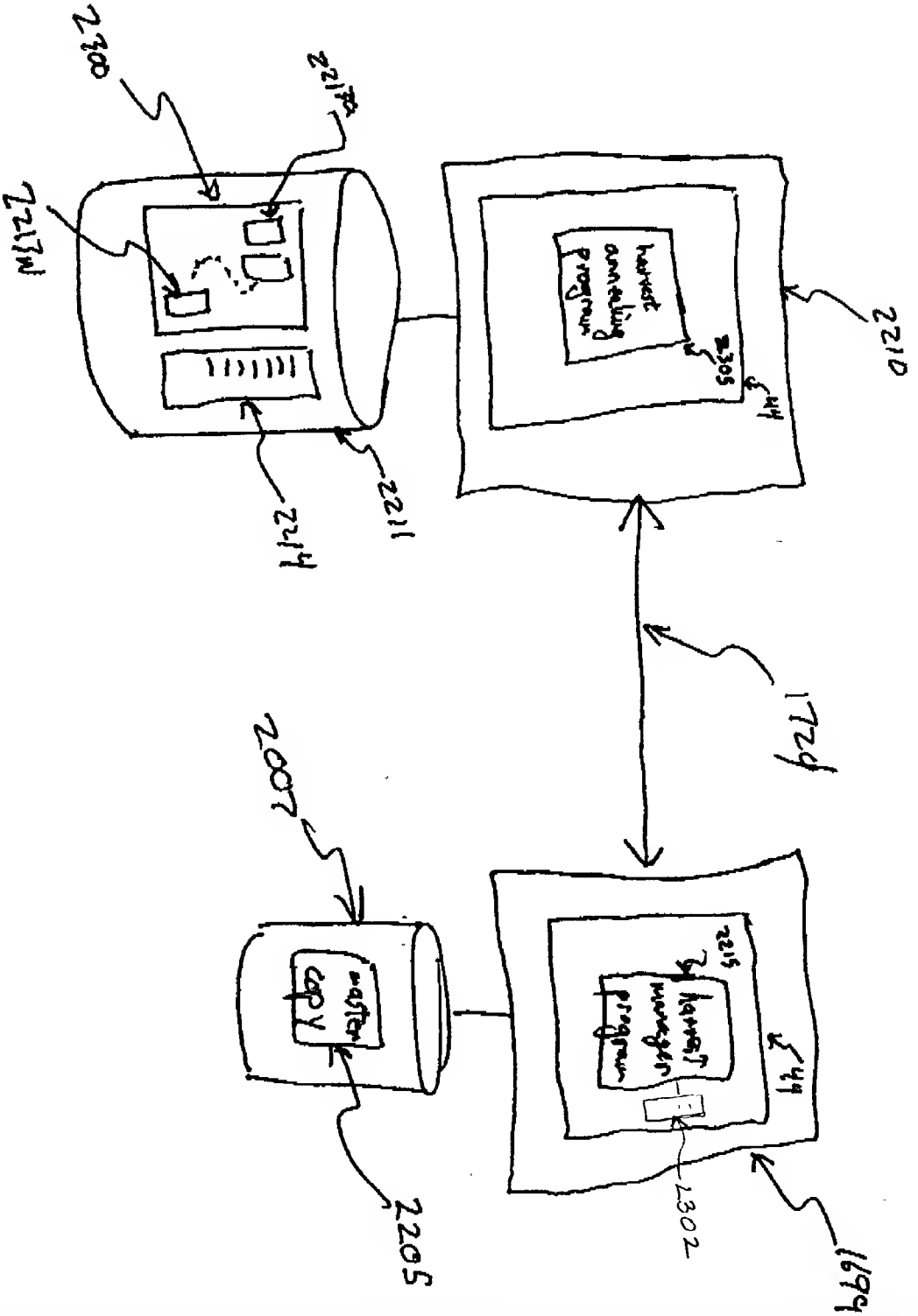

2300
2216
2217M
2214
2211
2210
harvest annealing program
305
44
1724
2007
master copy
2205
2215
harvest management program
44
1302
1694

FIG. 23A

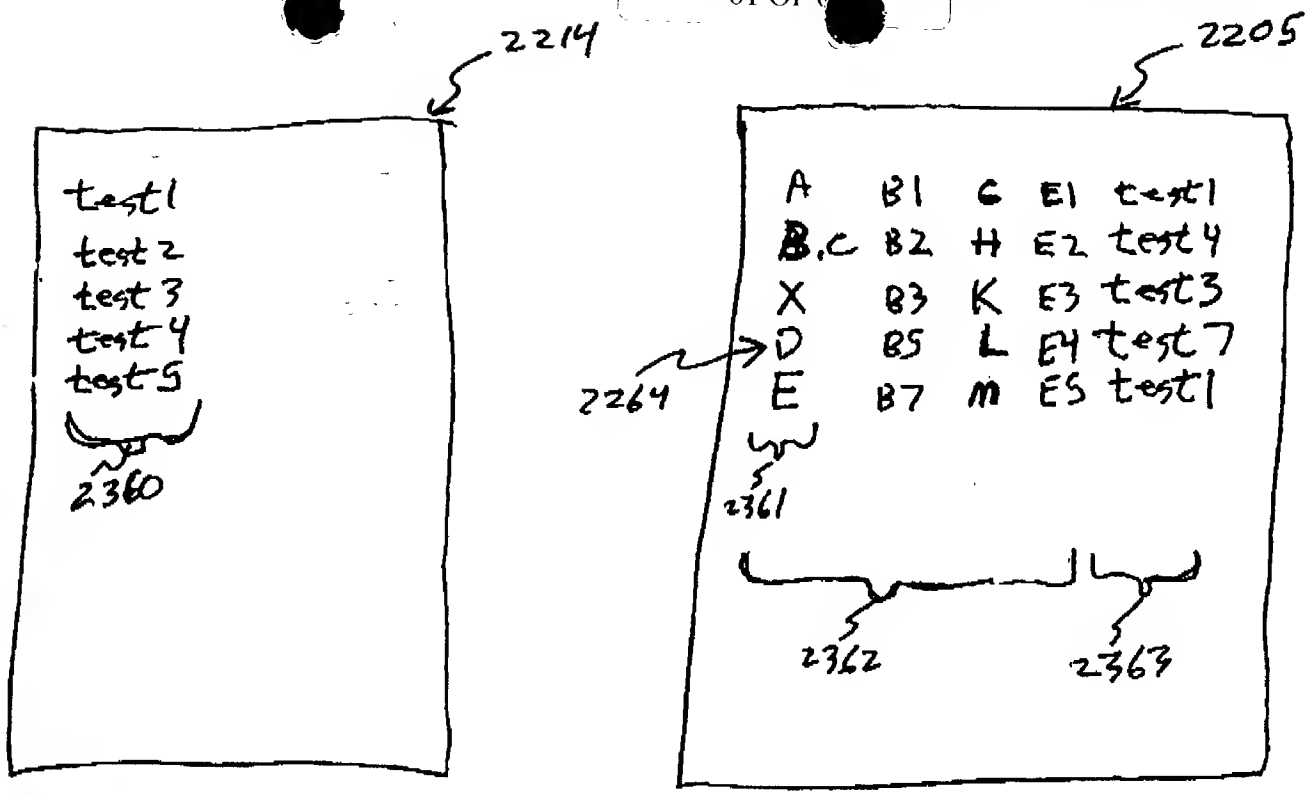

2214
test1
test2
test3
test4
test5
2360
2205
A B1 C E1 test1
B,C B2 H E2 test4
X B3 K E3 test3
D B5 L E4 test7
E B7 M E5 test1
2264
2361
2362
2363

FIG. 23B

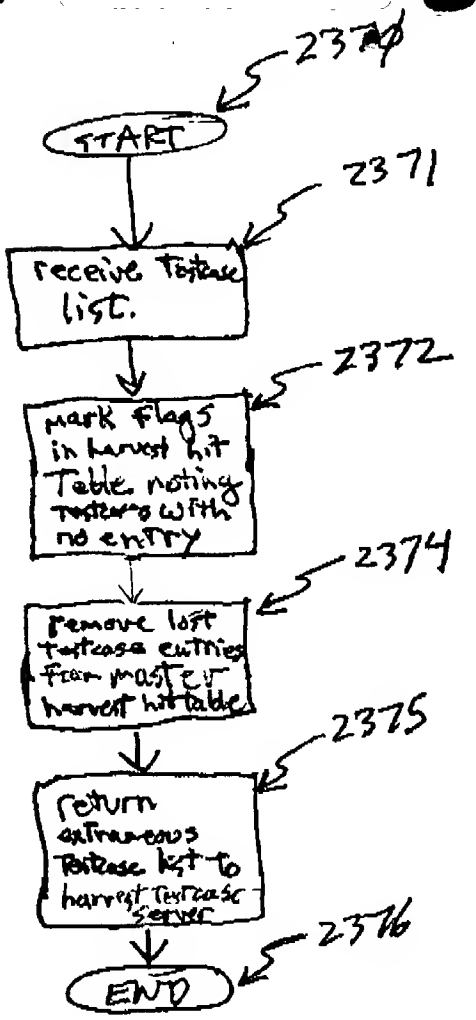

2370
START
2371
receive Testcase list.
2372
mark flags in harvest hit Table noting Testcases with no entry
2374
Remove left Testcase entries from master harvest hit table
2375
return extraneous Testcase list to harvest Testcase server
2376
END

FIG. 23C

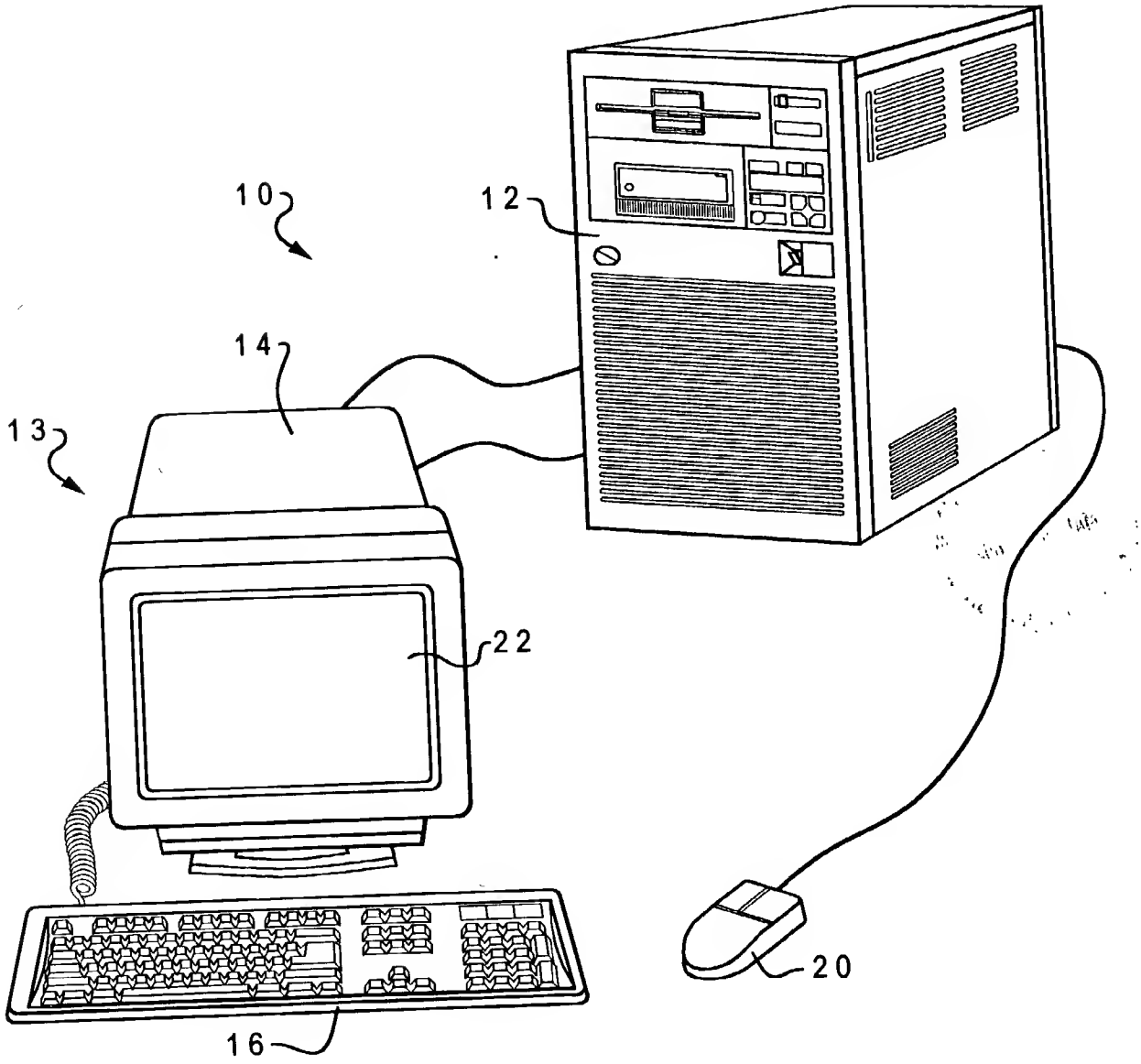


Fig. 1
Prior Art

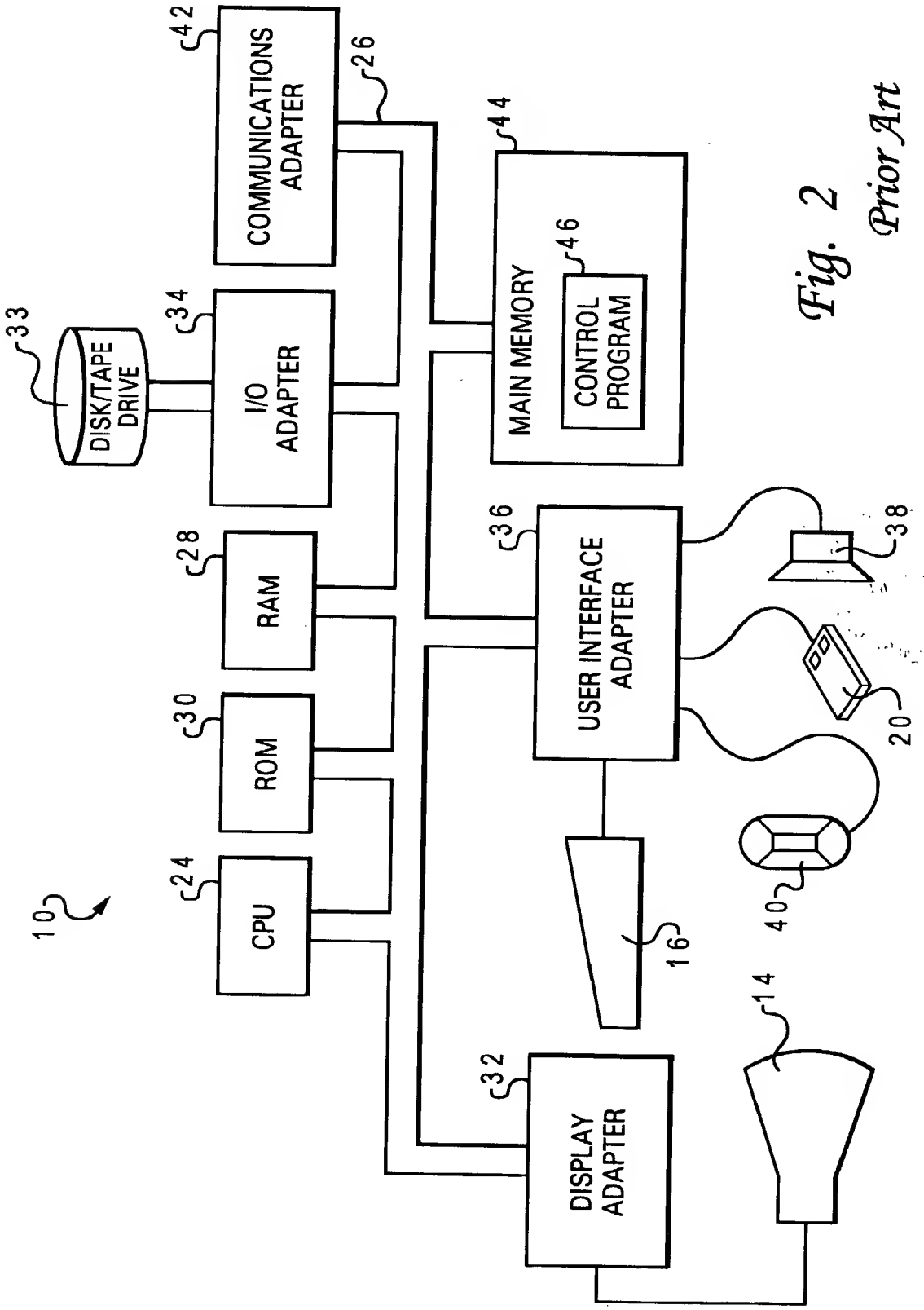


Fig. 2
Prior Art

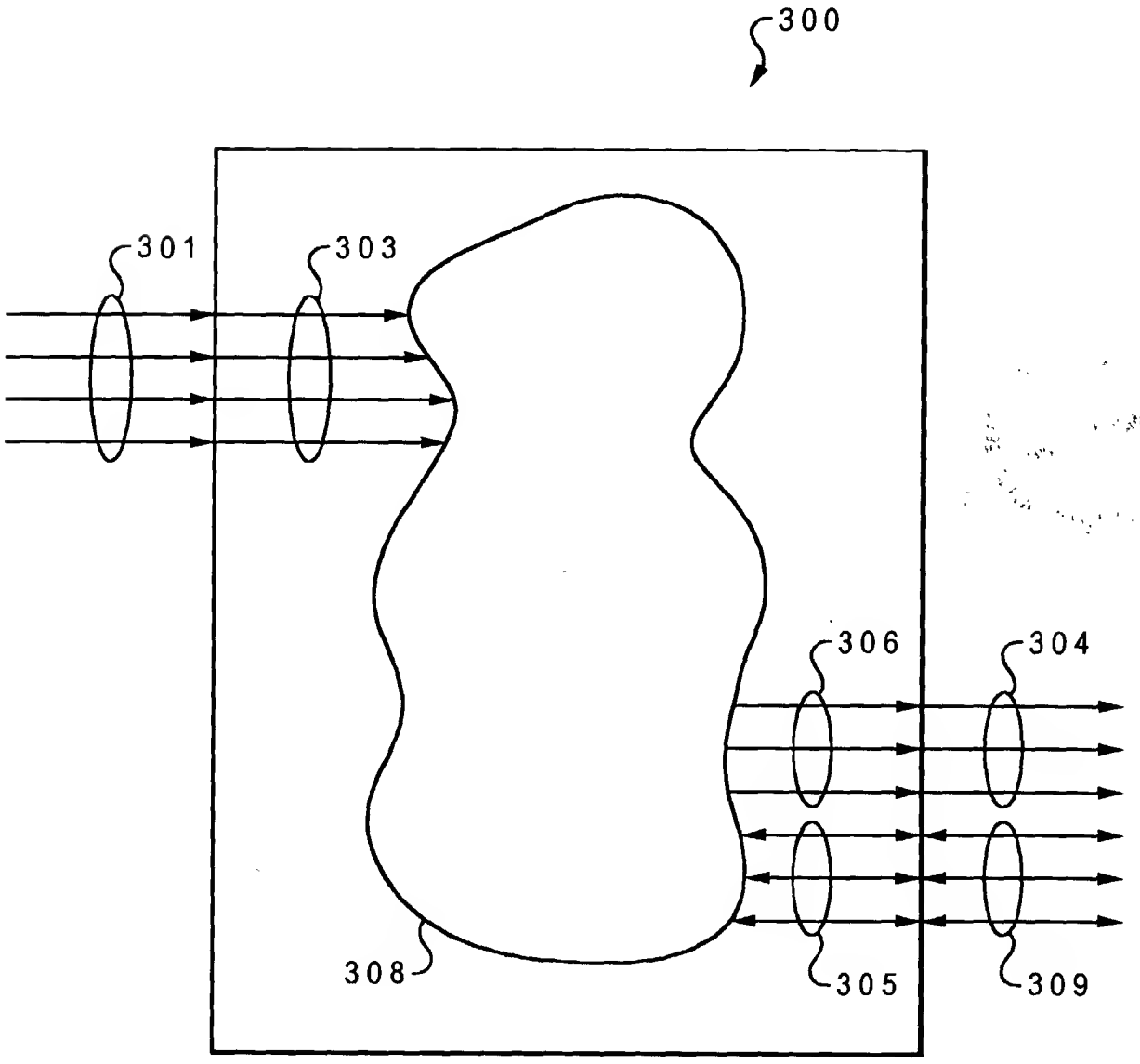


Fig. 3A

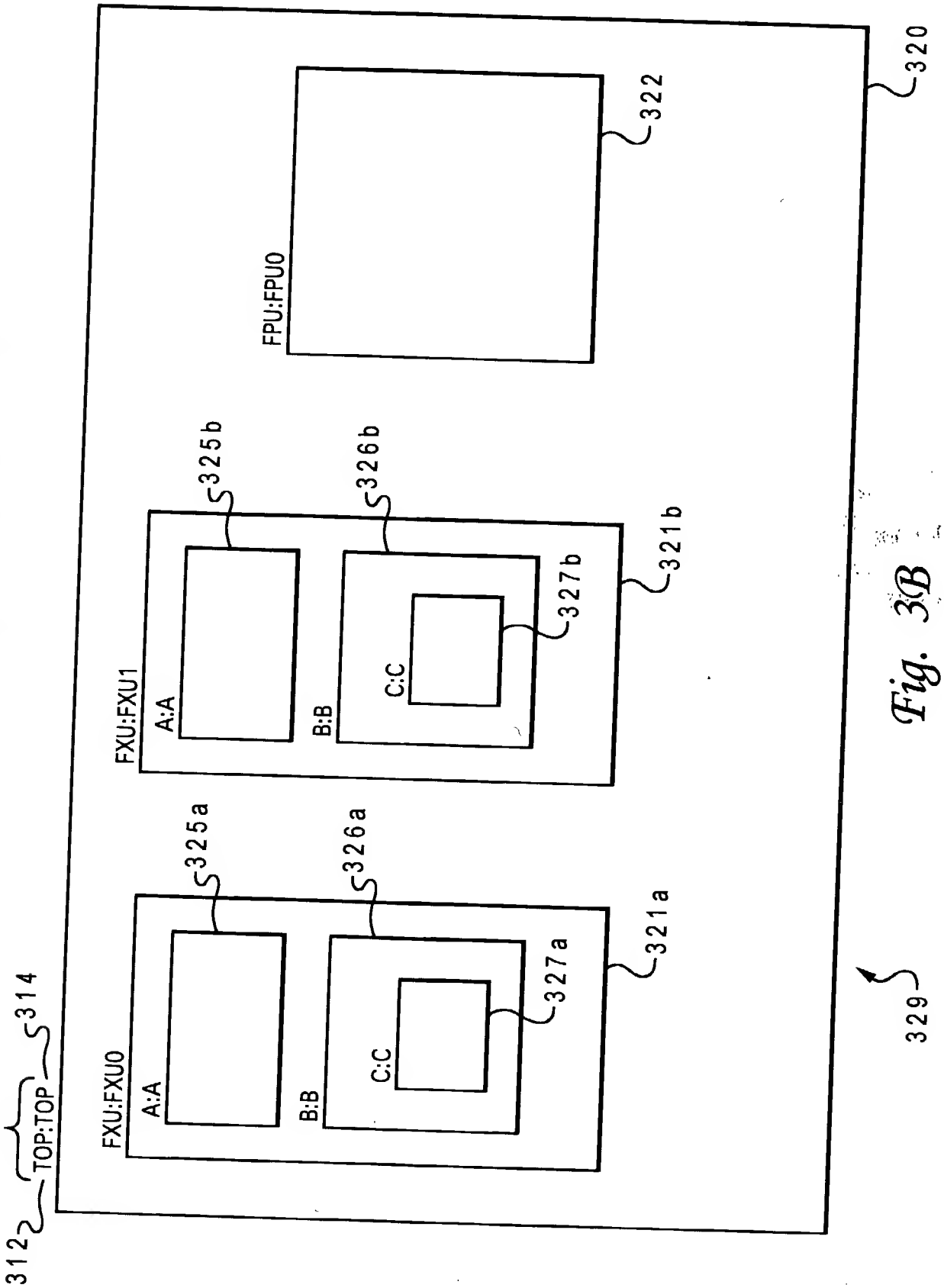

310
TOP:TOP
314
312
FXU:FXU0
A:A
B:B
C:C
325a
326a
327a
321a
FXU:FXU1
A:A
B:B
C:C
325b
326b
327b
321b
FPU:FPU0
322
320
329

Fig. 3B

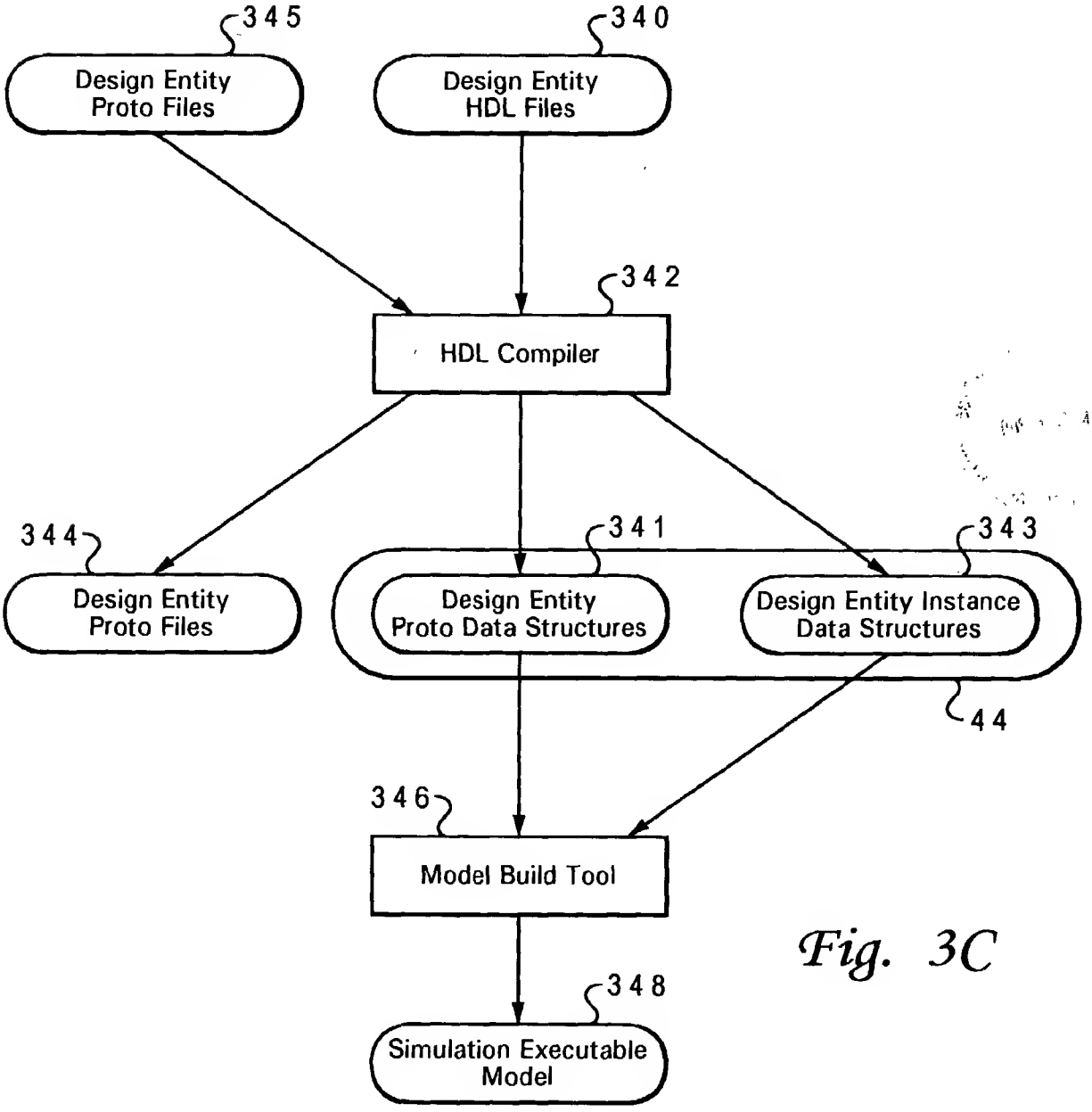

345
Design Entity Proto Files
340
Design Entity HDL Files
342
HDL Compiler
344
Design Entity Proto Files
341
Design Entity Proto Data Structures
343
Design Entity Instance Data Structures
44
346
Model Build Tool
348
Simulation Executable Model

Fig. 3C

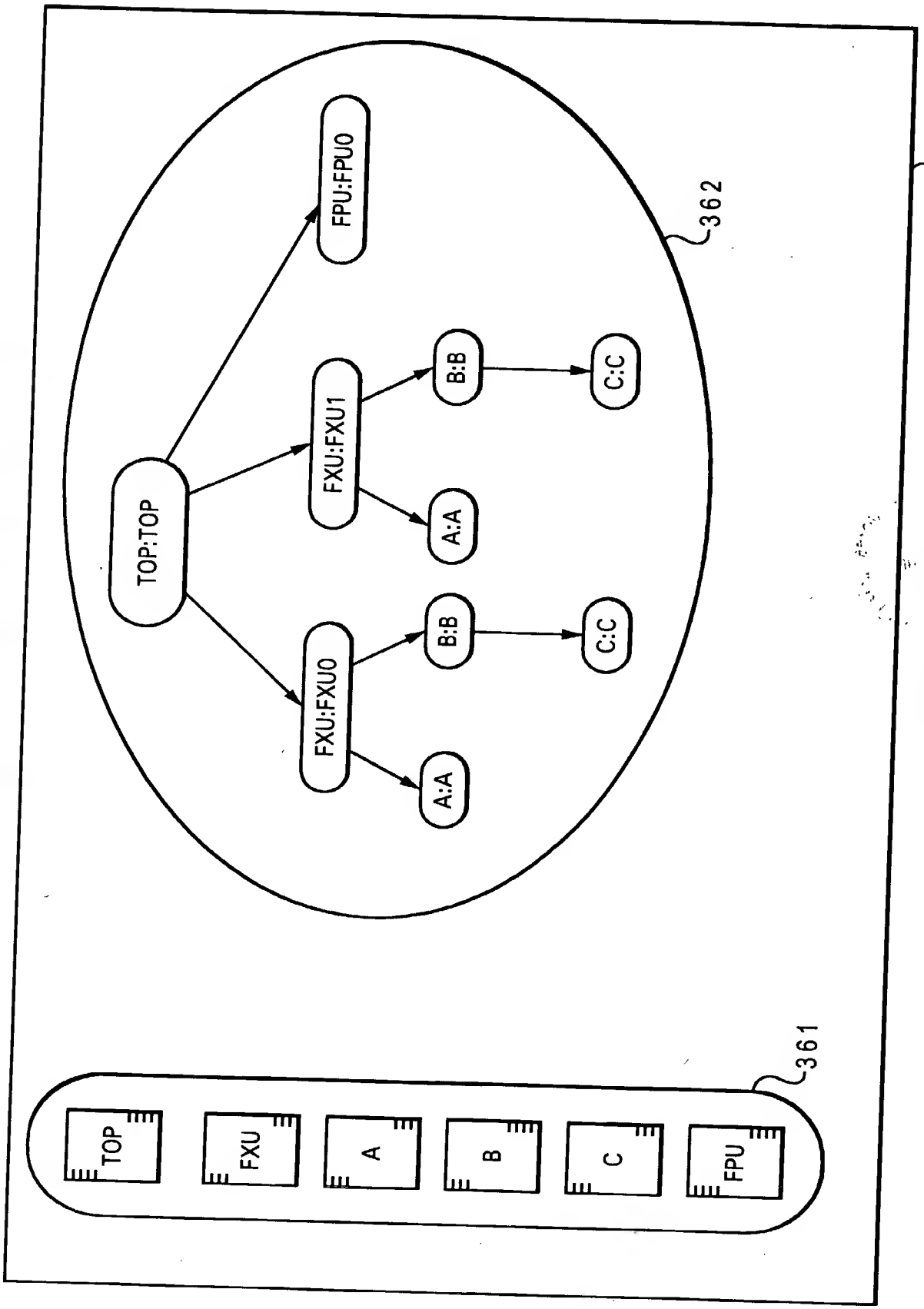

TOP
FXU
A
B
C
FPU
361
TOP:TOP
FPU:FPU0
FXU:FXU1
FXU:FXU0
A:A
B:B
C:C
A:A
B:B
C:C
362
44

Fig. 3D

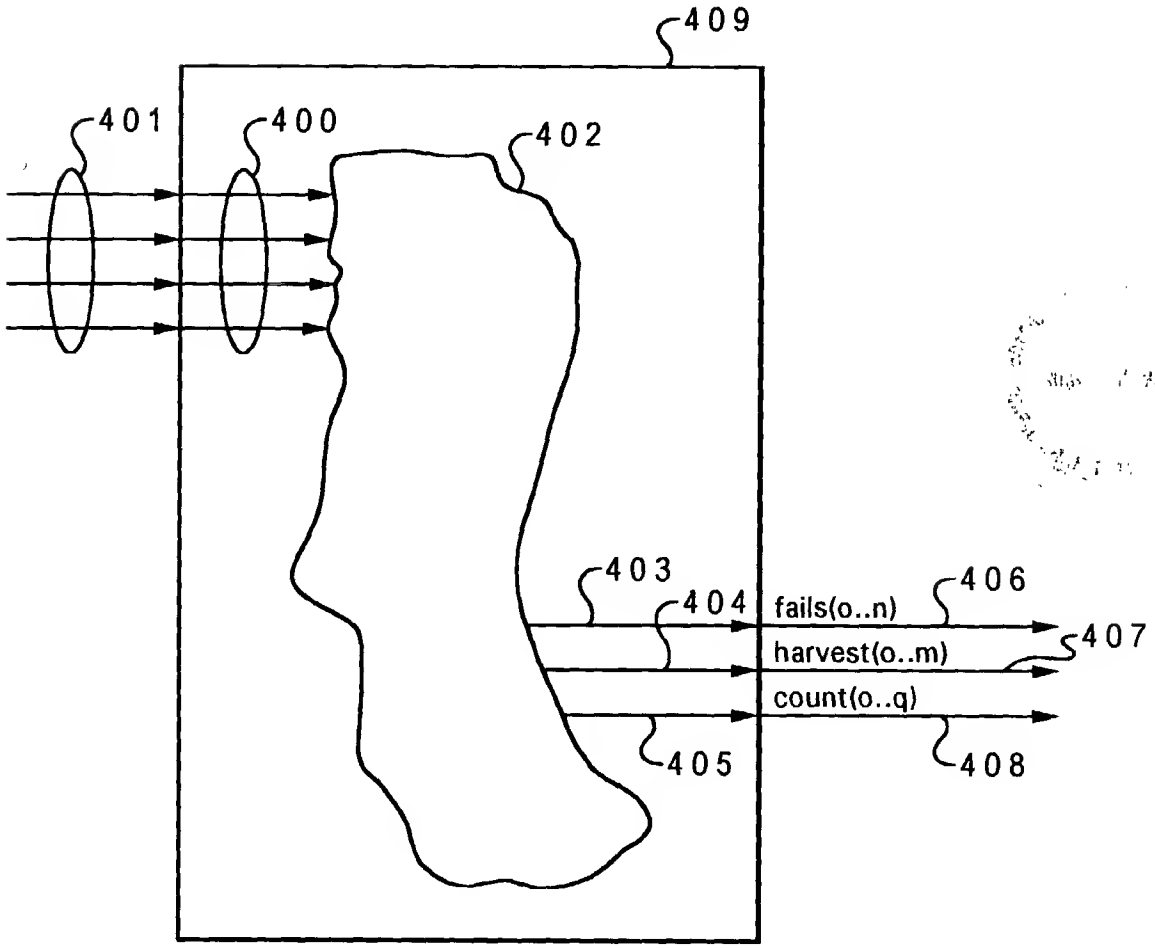

409
401
400
402
403
404
fails(o..n)
406
407
harvest(o..m)
count(o..q)
405
408

Fig. 4A

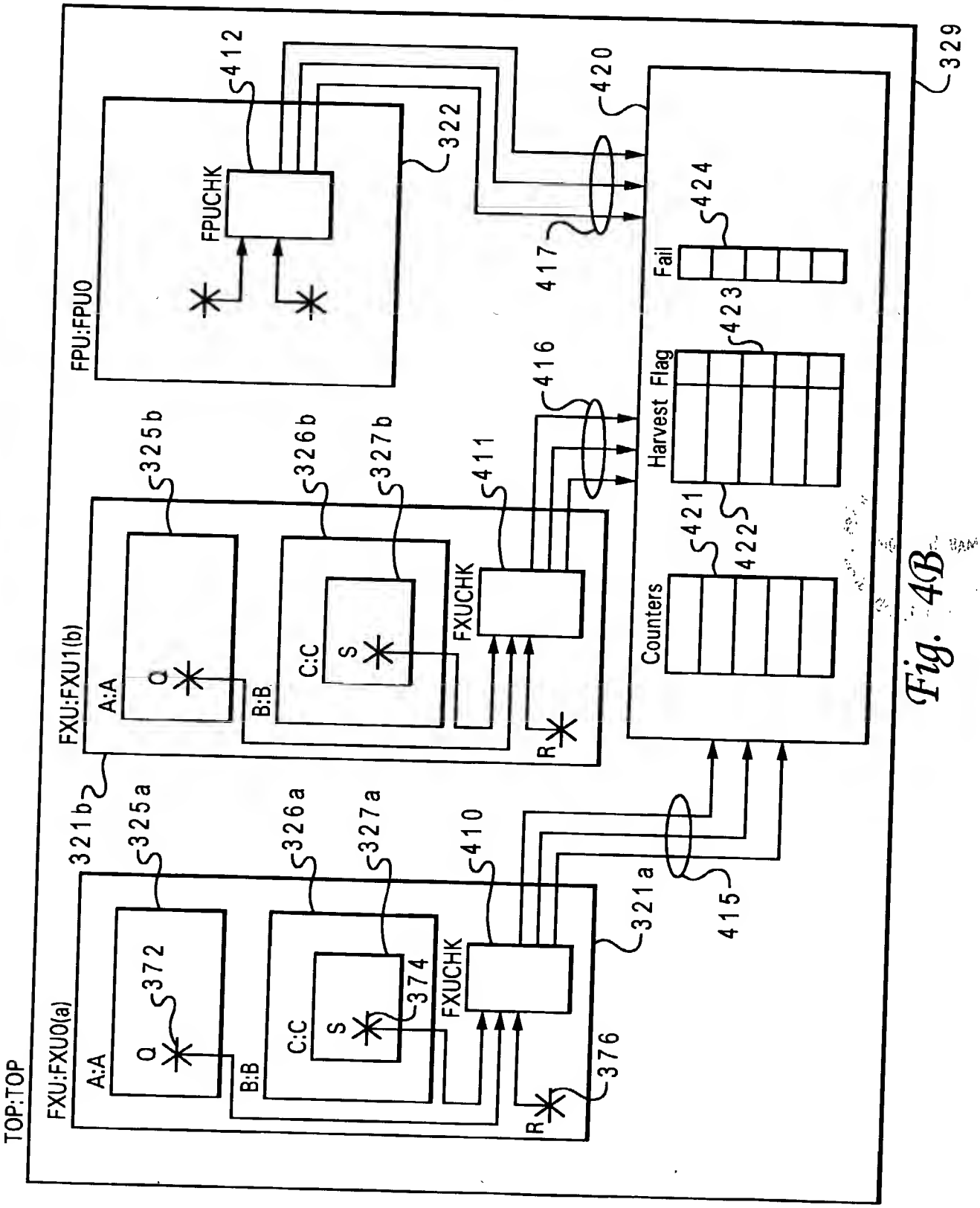

TOP:TOP
FXU:FXU0(a)
A:A
Q
372
B:B
C:C
S
374
FXUCHK
R
376
321a
325a
326a
327a
410
FXU:FXU1(b)
321b
325b
326b
327b
411
412
FPU:FPU0
FPUCHK
322
415
416
417
420
329
Counters
421
422
Harvest Flag
423
Fail
424

Fig. 4B

```
ENTITY FXUCHK IS

    PORT(   S_IN        :   IN std_ulogic;
            Q_IN        :   IN std_ulogic;
            R_IN        :   IN std_ulogic;
            clock       :   IN std_ulogic;
            fails       :   OUT std_ulogic_vector(0 to 1);
            counts      :   OUT std_ulogic_vector(0 to 2);
            harvests    :   OUT std_ulogic_vector(0 to 1);
        );

--!! BEGIN
--!! Design Entity: FXU;

--!! Inputs
--!! S_IN        =>      B.C.S;
--!! Q_IN        =>      A.Q;
--!! R_IN        =>      R;
--!! CLOCK       =>      clock;
--!! End Inputs

--!! Fail Outputs;
--!! 0 : "Fail message for failure event 0";
--!! 1 : "Fail message for failure event 1";
--!! End Fail Outputs;

--!! Count Outputs;
--!! 0 : <event0> clock;
--!! 1 : <event1> clock;
--!! 2 : <event2> clock;
--!! End Count Outputs;

--!! Harvest Outputs;
--!! 0 : "Message for harvest event 0";
--!! 1 : "Message for harvest event 1";
--!! End Harvest Outputs;

--!! End;

ARCHITECTURE example of FXUCHK IS

BEGIN

    ... HDL code for entity body section ...

END;
```

450 452 453 454 455 456 457 451 458 440

Fig. 4C

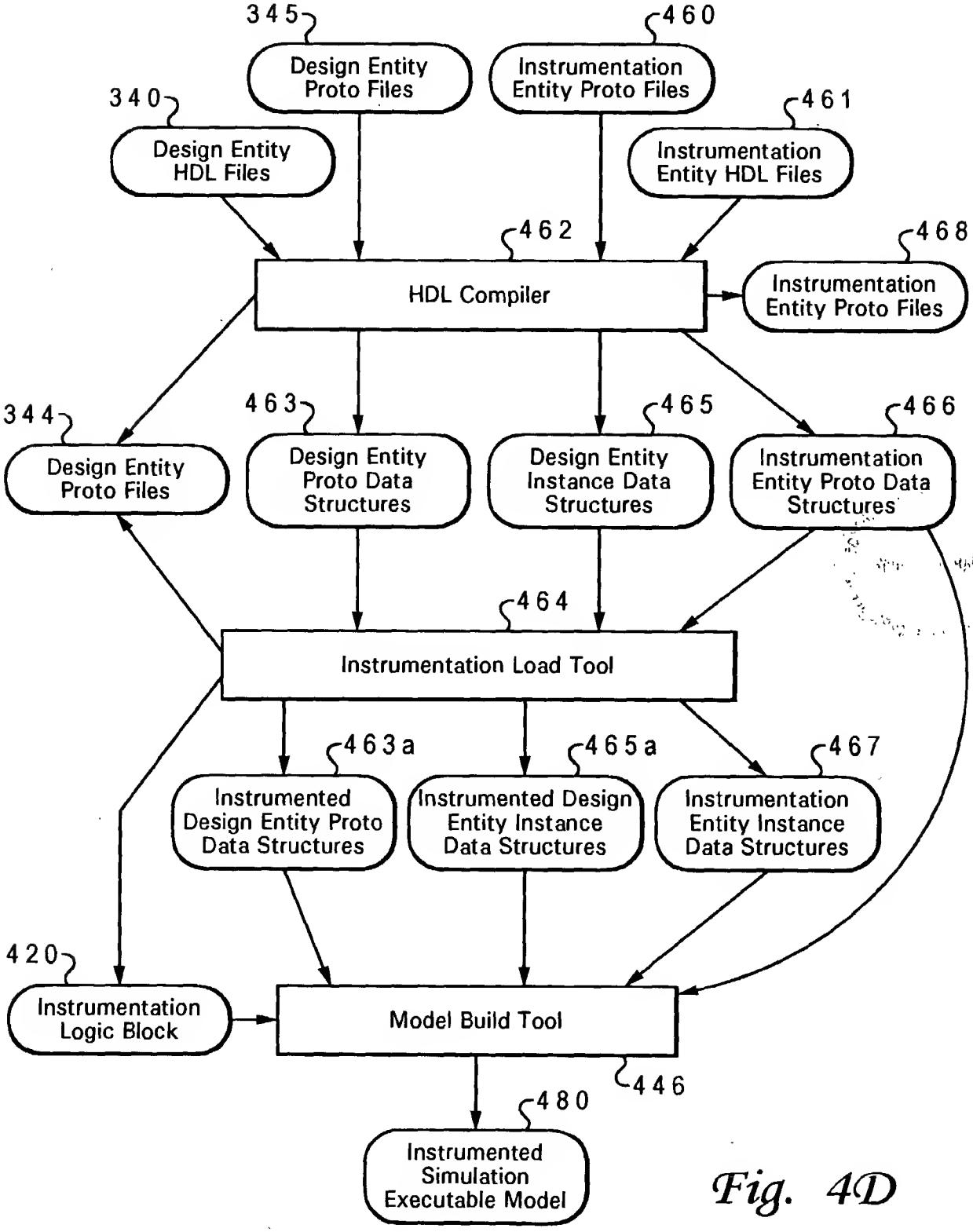

345
Design Entity Proto Files
460
Instrumentation Entity Proto Files
340
Design Entity HDL Files
461
Instrumentation Entity HDL Files
462
HDL Compiler
468
Instrumentation Entity Proto Files
344
Design Entity Proto Files
463
Design Entity Proto Data Structures
465
Design Entity Instance Data Structures
466
Instrumentation Entity Proto Data Structures
464
Instrumentation Load Tool
463a
Instrumented Design Entity Proto Data Structures
465a
Instrumented Design Entity Instance Data Structures
467
Instrumentation Entity Instance Data Structures
420
Instrumentation Logic Block
Model Build Tool
446
480
Instrumented Simulation Executable Model

Fig. 4D

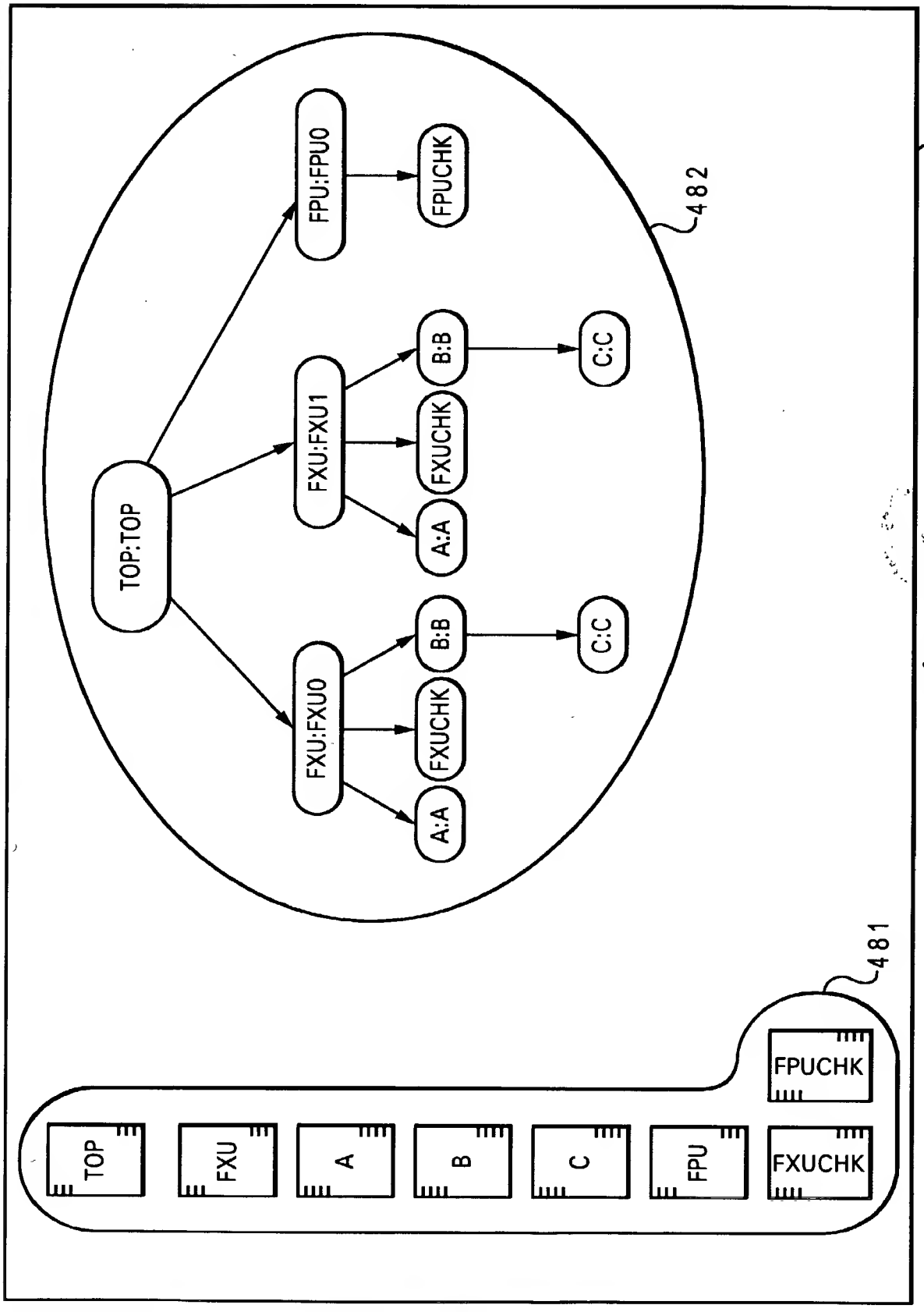

44
482
481
TOP:TOP
FPU:FPU0
FPUCHK
FXU:FXU1
B:B
FXUCHK
A:A
C:C
FXU:FXU0
B:B
FXUCHK
A:A
C:C
TOP
FXU
A
B
C
FPU
FXUCHK
FPUCHK

Fig. 4E

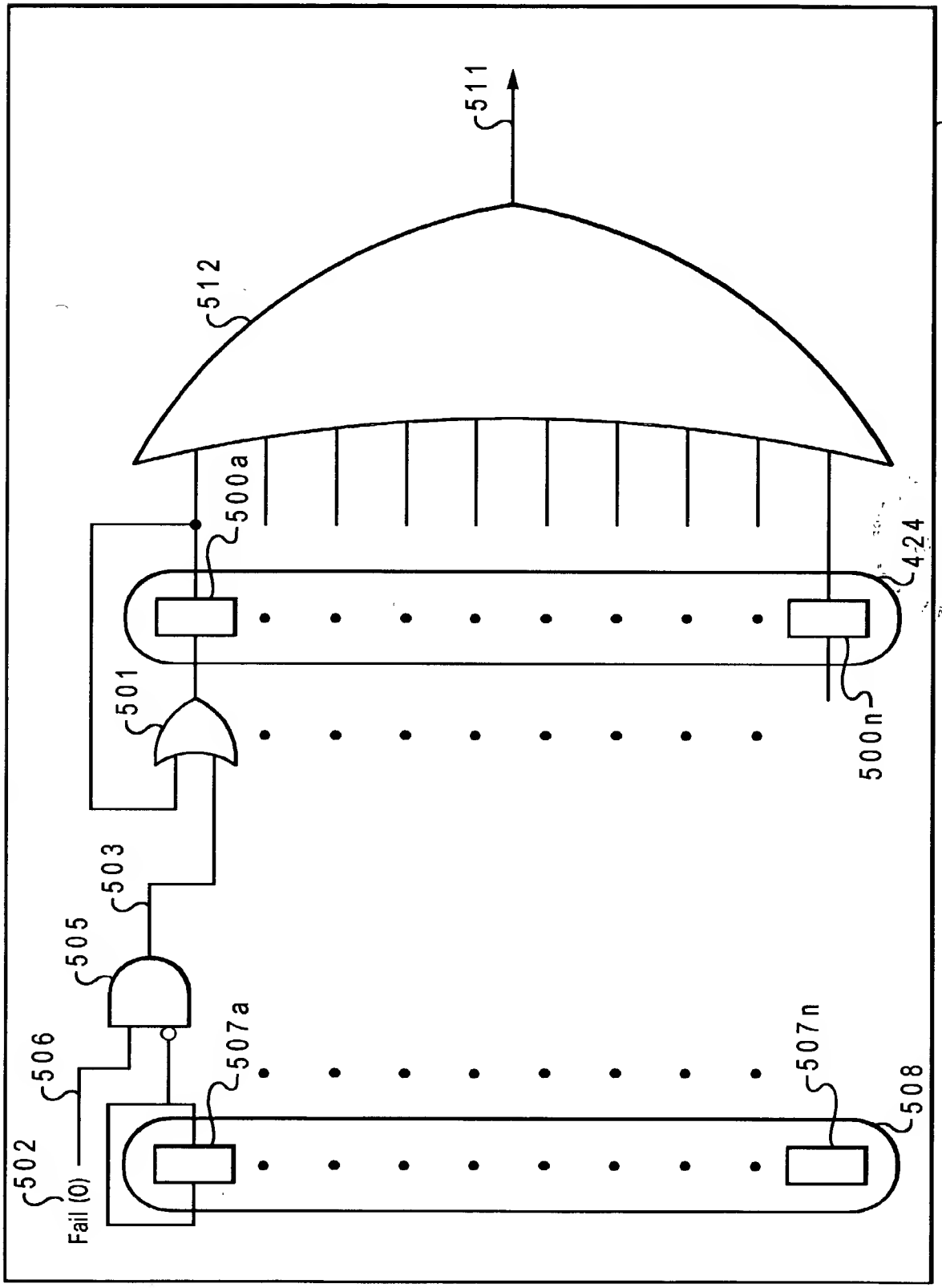

420
511
512
500a
424
501
500n
503
505
506
507a
507n
508
502
Fail (0)

Fig. 5A

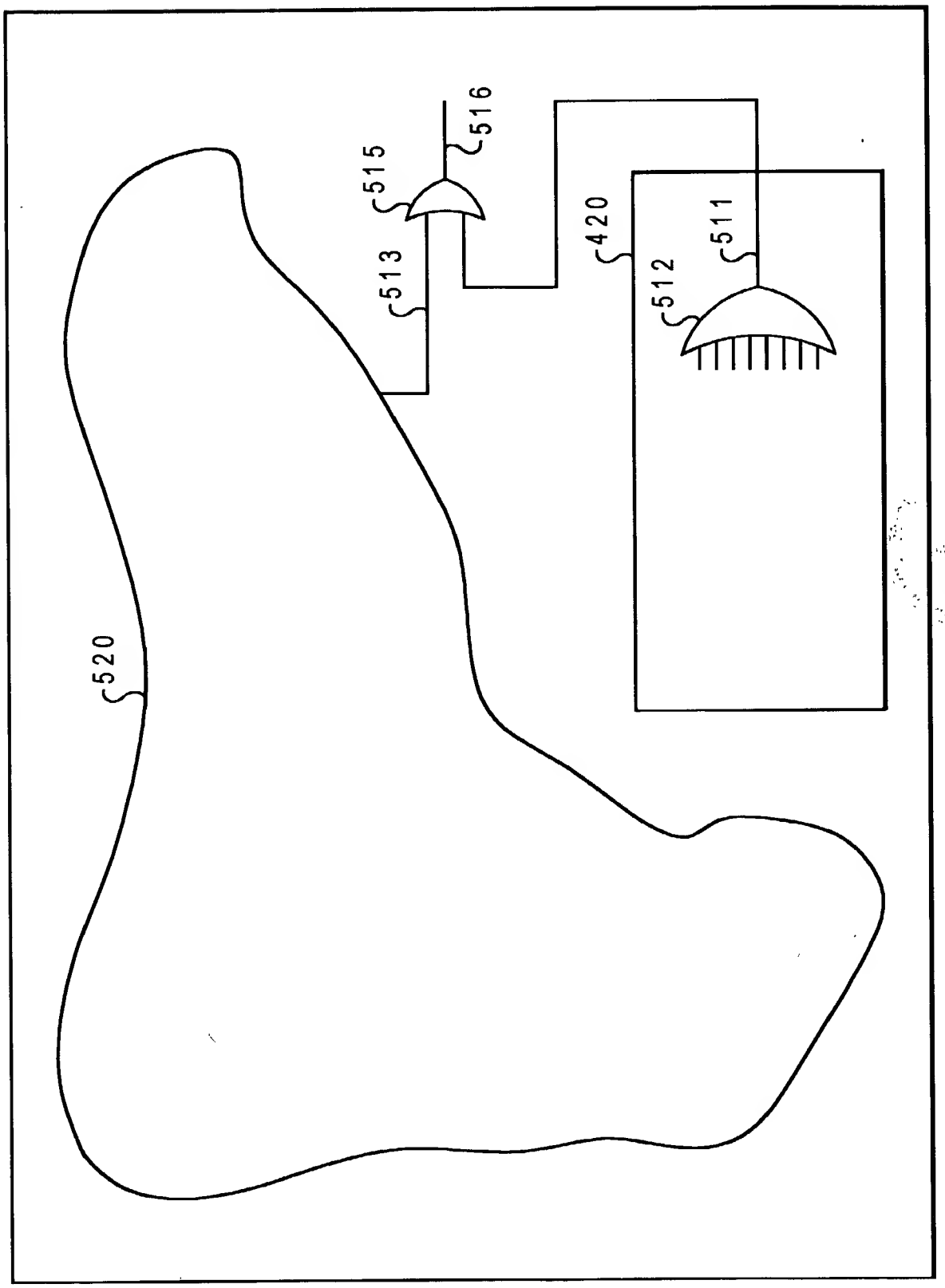


Fig. 5B

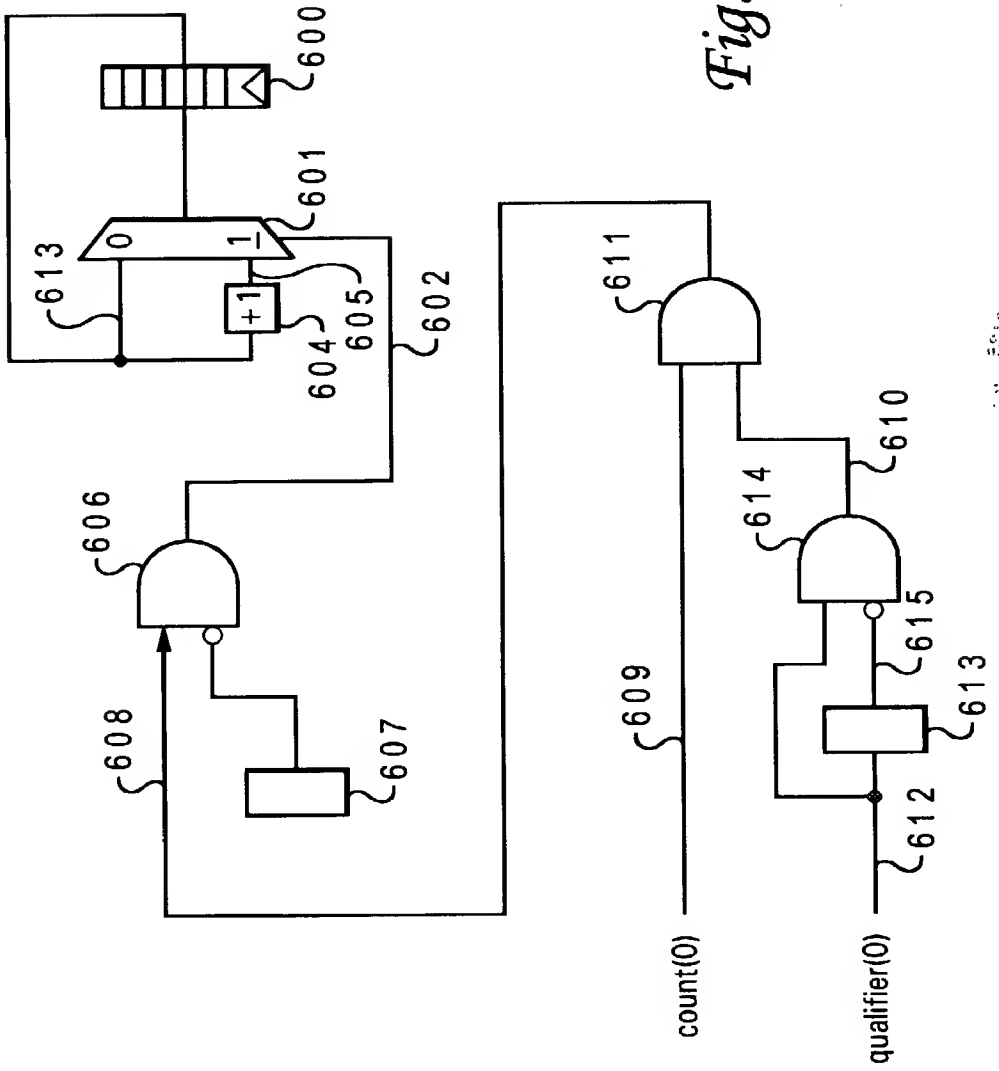

600
601
0
1
613
+1
604
605
602
606
608
607
611
610
614
615
613
609
612
count(0)
qualifier(0)

Fig. 6A

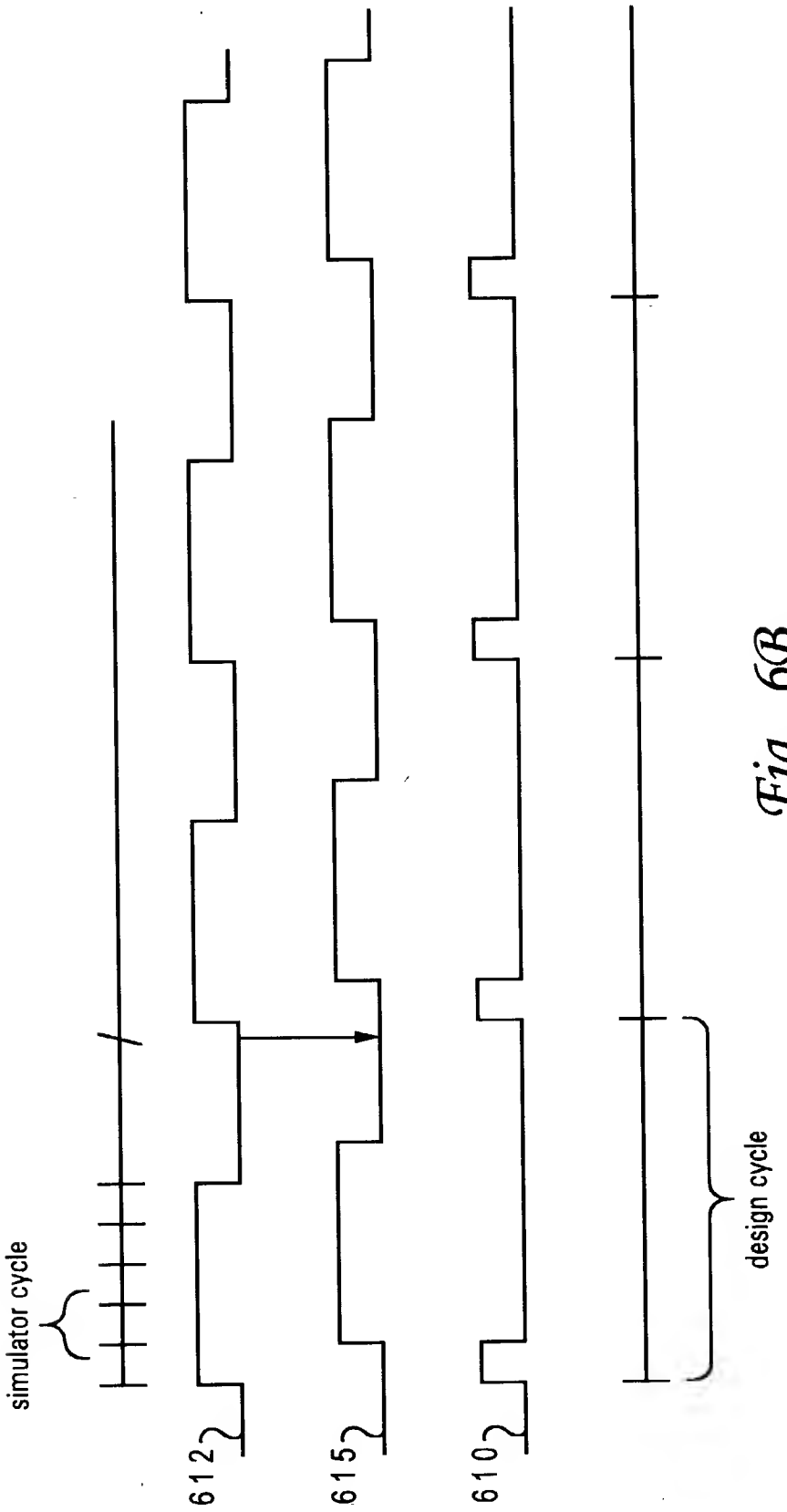


Fig. 6B

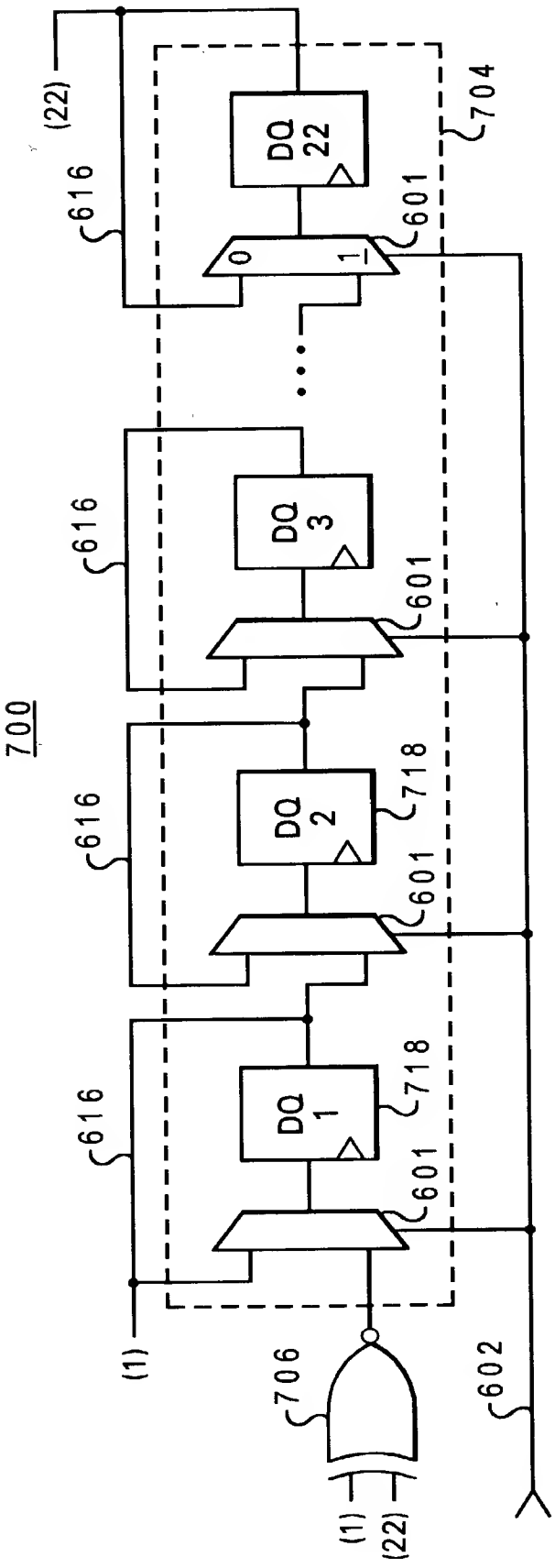


Fig. 7

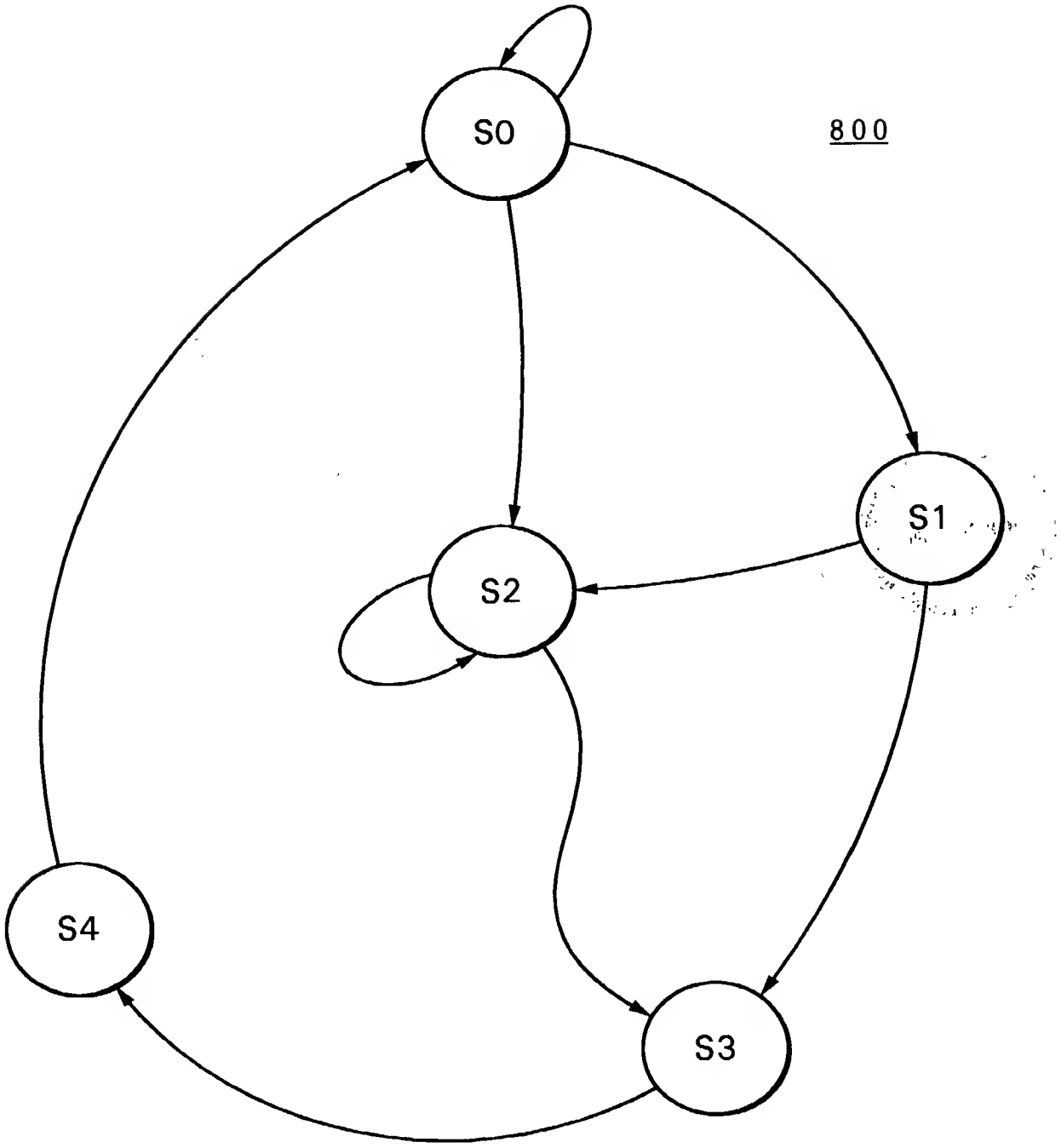


Fig. 8A
Prior Art

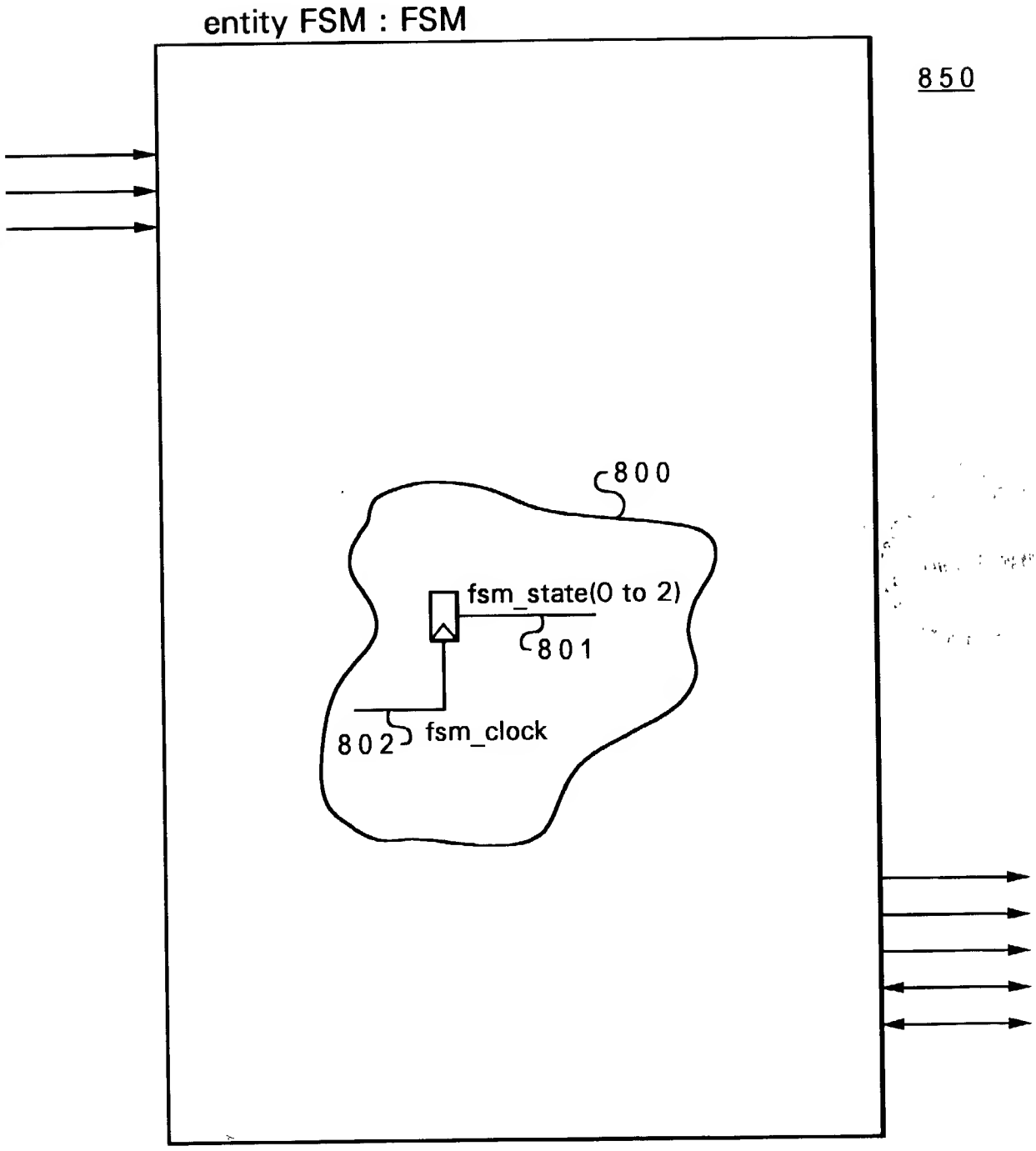


Fig. 8B
Prior Art

```
ENTITY FSM IS

    PORT(
            ....ports for entity fsm....
        );

    ARCHITECTURE FSM OF FSM IS

    BEGIN

            ... HDL code for FSM and rest of the entity ...

            fsm_state(0 to 2) <= ... Signal 801 ...

853 --!! Embedded FSM : examplefsm;
859 --!! clock        : (fsm_clock);
854 --!! state_vector : (fsm_state(0 to 2));
855 --!! states       : (S0, S1, S2, S3, S4);
856 --!! state_encoding : ('000', '001', '010', '011', '100');
857 --!! arcs         : (S0 => S0, S0 => S1, S0 => S2,
    --!!               (S1 => S2, S1 => S3, S2 => S2,
    --!!               (S2 => S3, S3 => S4, S4 => S0);
858 --!! End FSM;

                    ,
                    ,
                    ,
                    ,
                    ,
                    ,
                    ,

    END;
```

852 860

Fig. 8C

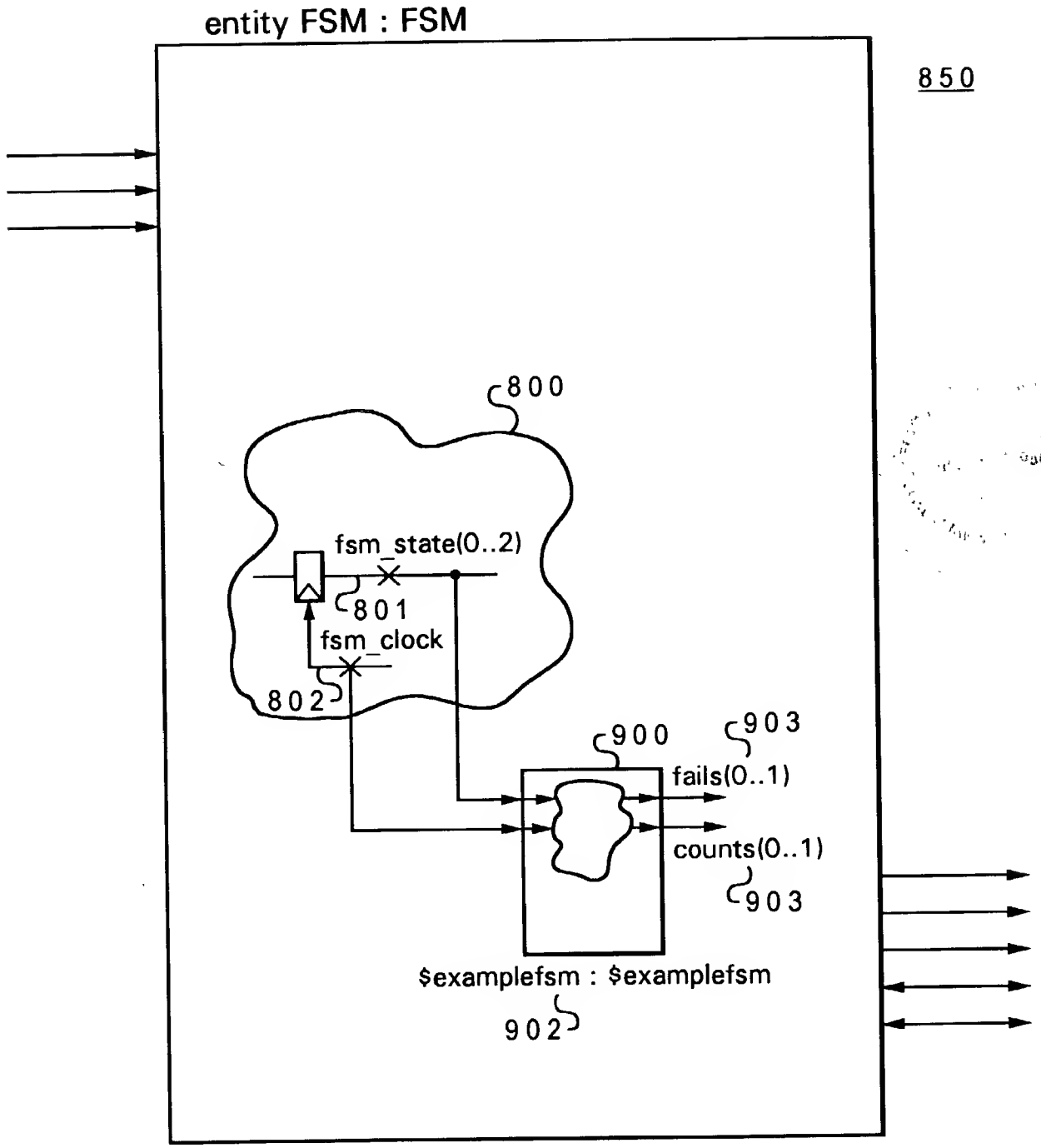

entity FSM : FSM
850
800
fsm_state(0..2)
801
fsm_clock
802
900
903
fails(0..1)
counts(0..1)
903
$examplefsm : $examplefsm
902

Fig. 9

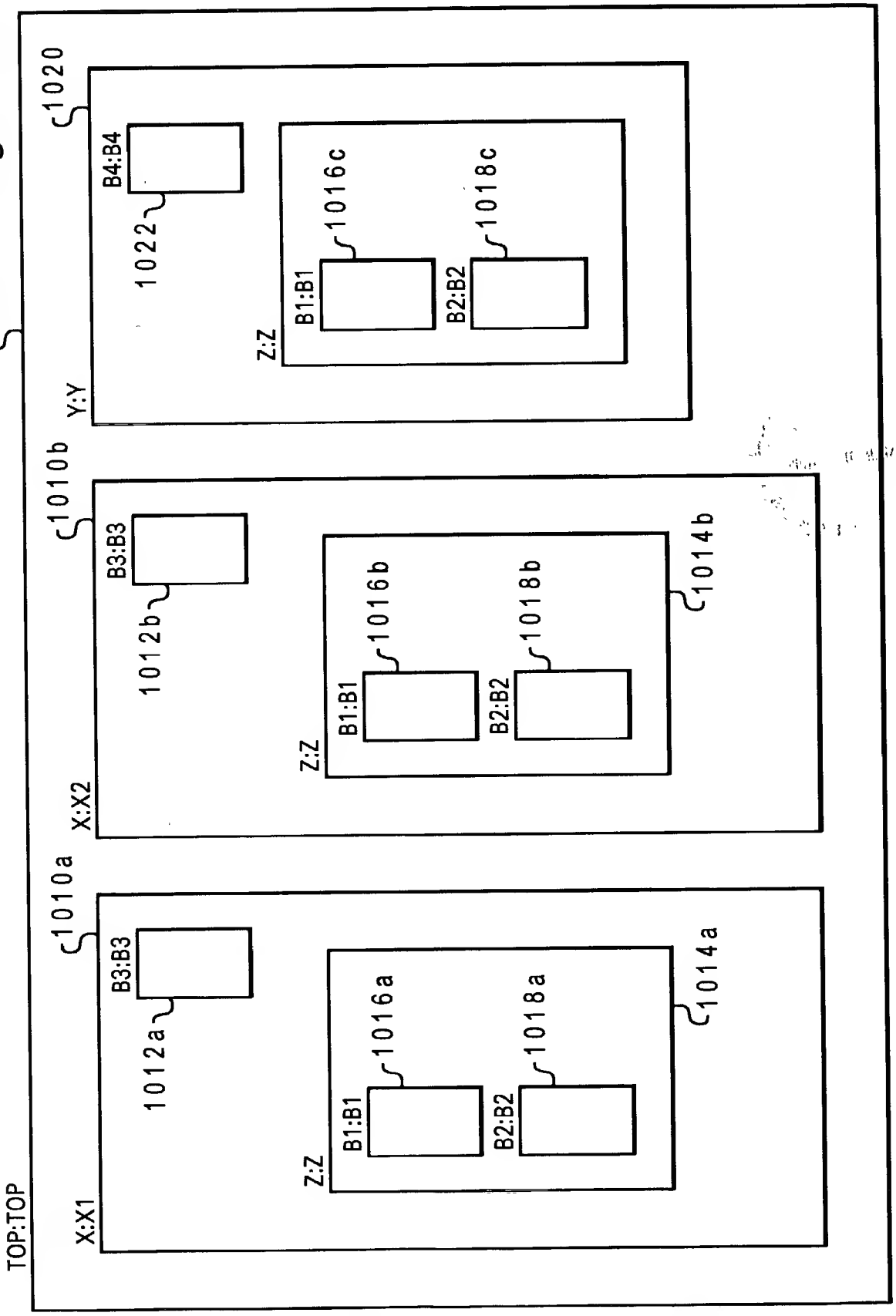

1000
TOP:TOP
X:X1
1010a
B3:B3
1012a
Z:Z
B1:B1
1016a
B2:B2
1018a
1014a
X:X2
1010b
B3:B3
1012b
Z:Z
B1:B1
1016b
B2:B2
1018b
1014b
Y:Y
1020
B4:B4
1022
Z:Z
B1:B1
1016c
B2:B2
1018c

Fig. 10A

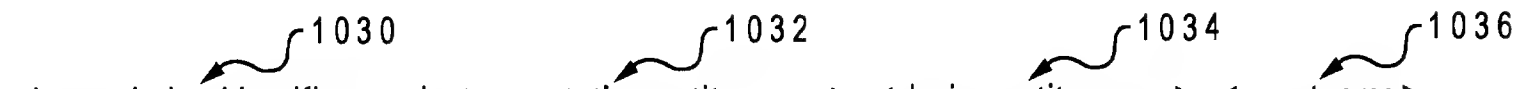


Fig. 10B

1030	1032	1034	1036	
X1	B3	X	COUNT1	1040
X1.Z	B1	Z	COUNT1	1041
X1.Z	B2	Z	COUNT1	1042
X2	B3	X	COUNT1	1043
X2.Z	B1	Z	COUNT1	1044
X2.Z	B2	Z	COUNT1	1045
Y	B4	Y	COUNT1	1046
Y.Z	B1	Z	COUNT1	1047
Y.Z	B2	Z	COUNT1	1048

Fig. 10C

Fig. 10D

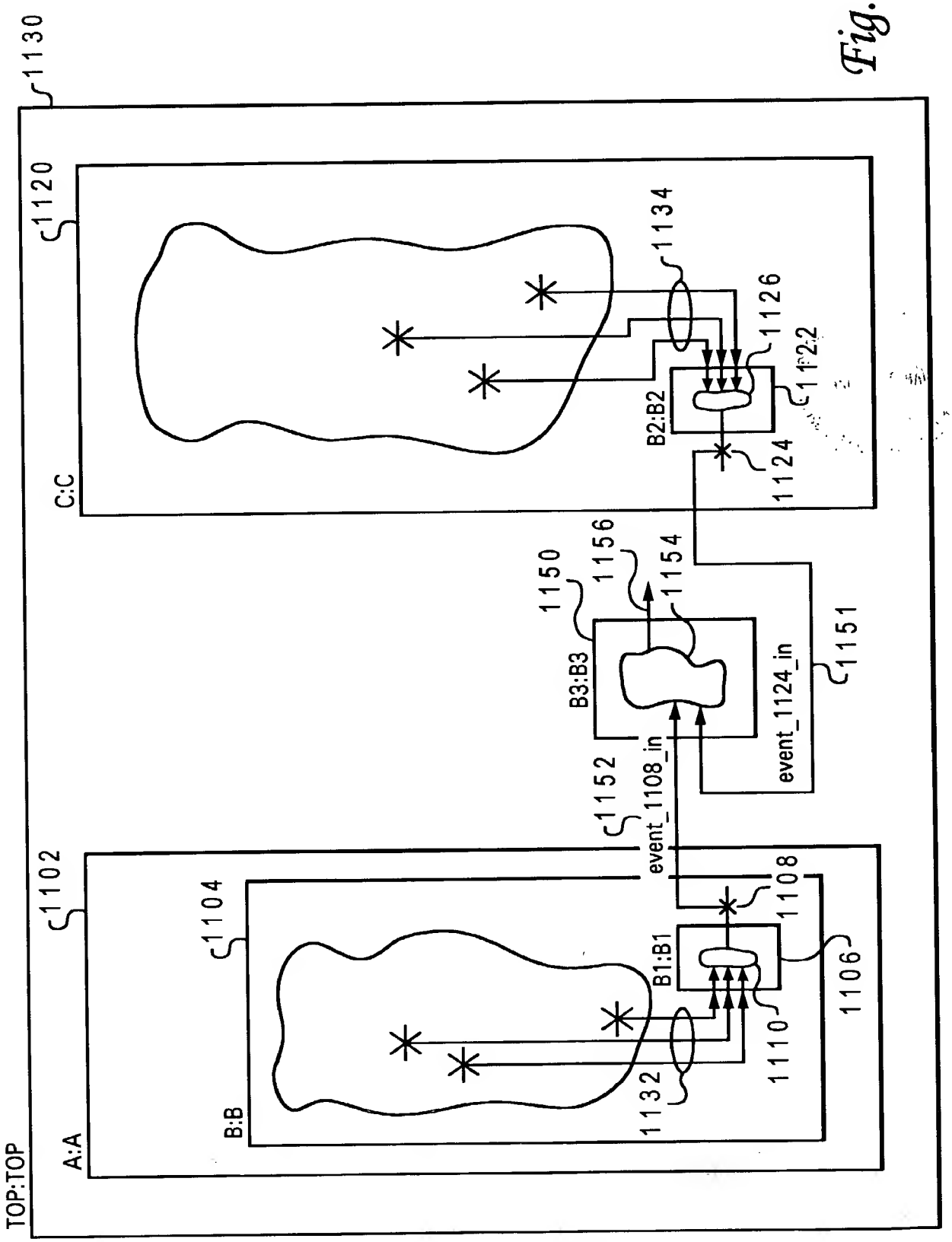

TOP:TOP
1130
A:A
1102
B:B
1104
C:C
1120
1132
1110
B1:B1
1106
1108
event_1108_in
1152
1150
B3:B3
1156
1154
event_1124_in
1151
1134
1126
B2:B2
1122
1124

Fig. 11A

```
--!! Inputs
--!! event_1108_in < = C.[B2.count.event_1108];
--!! event_1124_in < = A.B.[B1.count.event_1124];
--!! End Inputs
```

1163 1165 1161 1162 1164 1166

Fig. 11B

```
--!! Inputs
--!! event_1108_in < = C.[count.event_1108];
--!! event_1124_in < = B.[count.event_1124];
--!! End Inputs
```

1171 1172

Fig. 11C

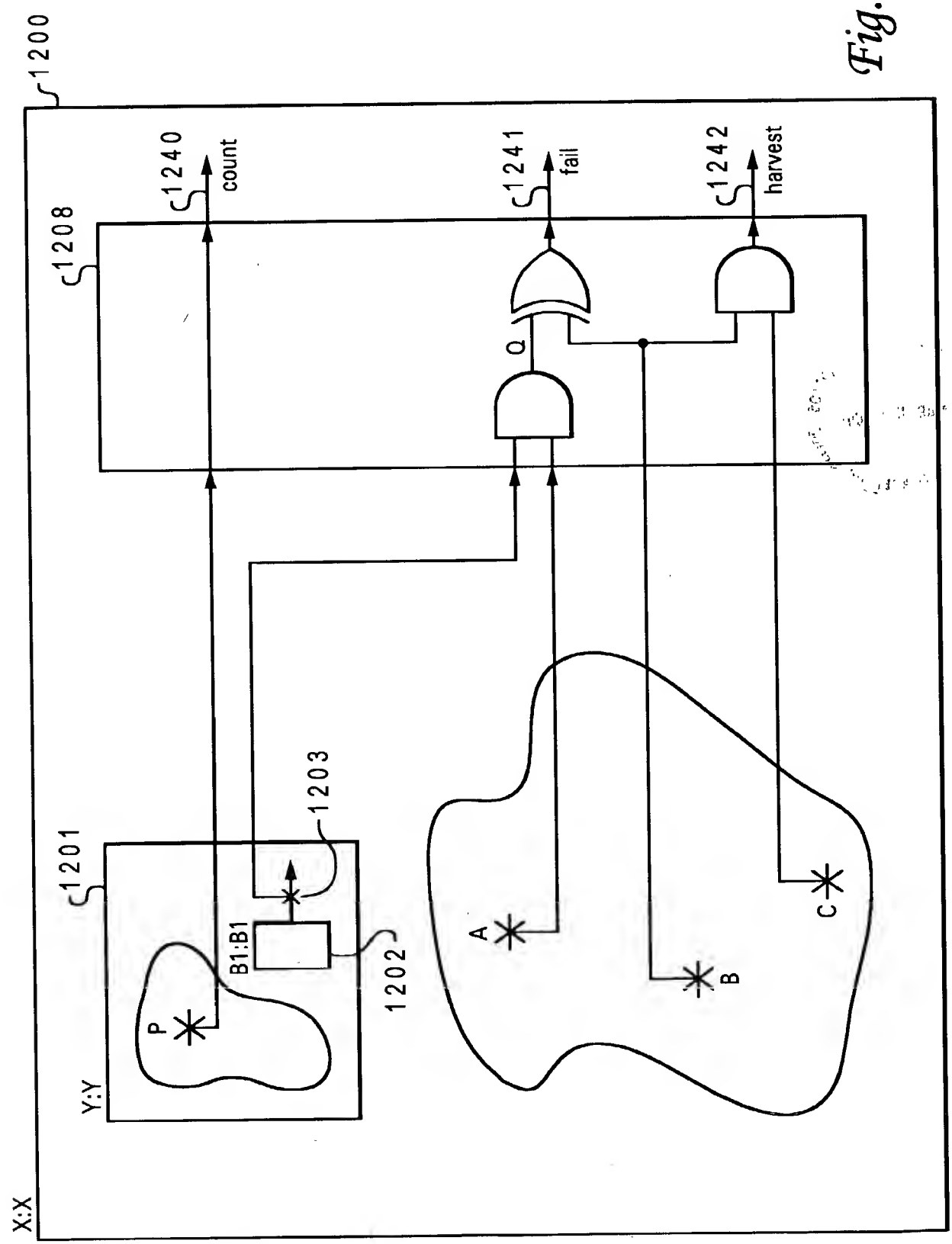

1200
1208
1240
count
1241
fail
1242
harvest
Q
1201
1203
B1:B1
1202
P
Y:Y
A
B
C
X:X

Fig. 12A

```
ENTITY X IS

      PORT(     :
                :
                :
           );

ARCHITECTURE example of X IS

BEGIN
      .
      .
      .
      .
      .
   ... HDL code for X ...
      .
      .
      .
      .
      .

1221  Y:Y
      PORT MAP(   :
                  :
               );

1222  A <= ....
      B <= ....
      C <= ....

1223  --!! [count, countname0, clock] <= Y.P;          1230 1232
      --!! Q <= Y. [B1.count.count1] AND A;                  1234
      --!! [fail, failname0, "fail msg"] <= Q XOR B;
      --!! [harvest, harvestname0, "harvest msg"] <= B AND C;
                                                        1236
END;
```

1220

Fig. 12B

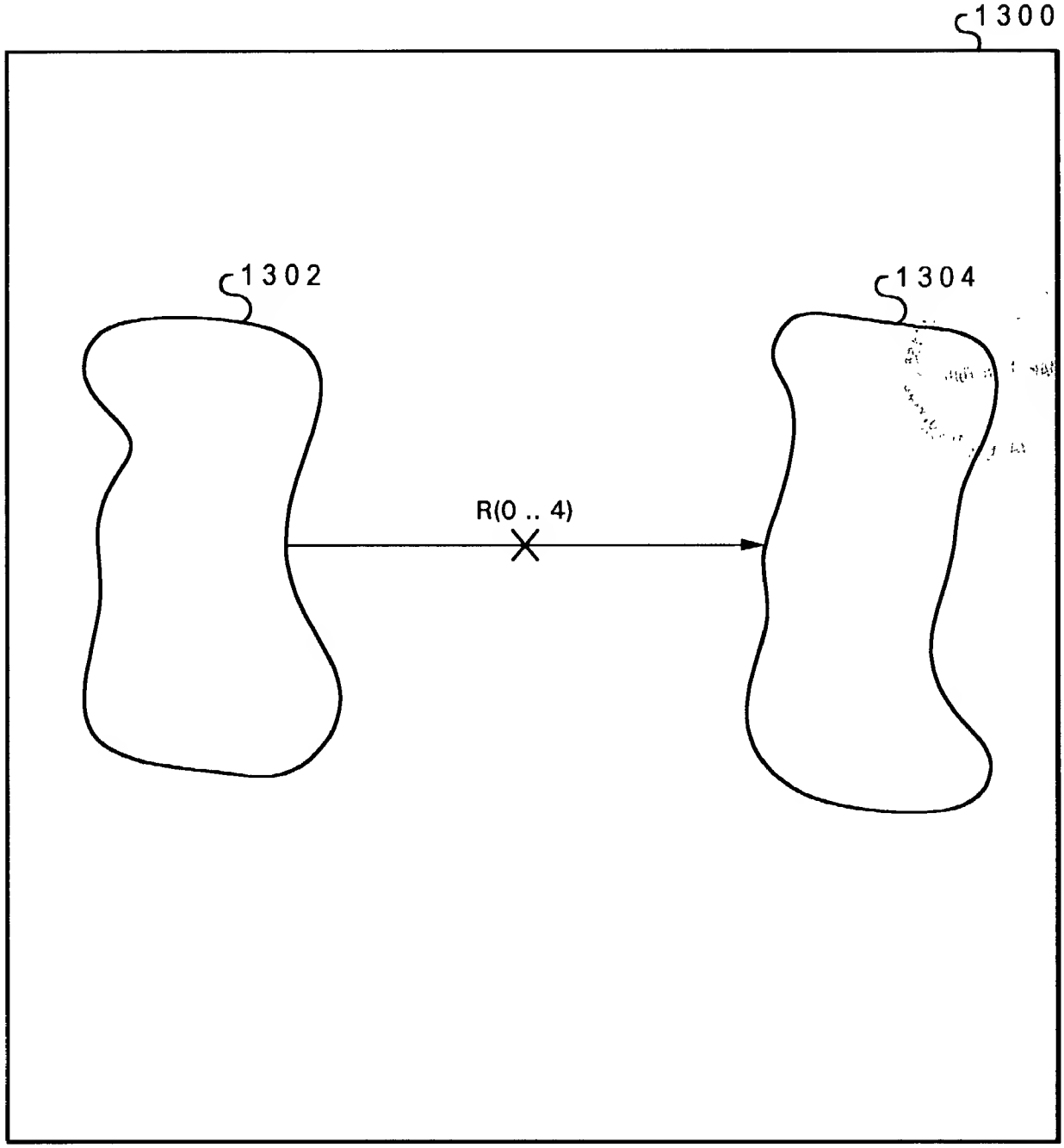


Fig. 13A

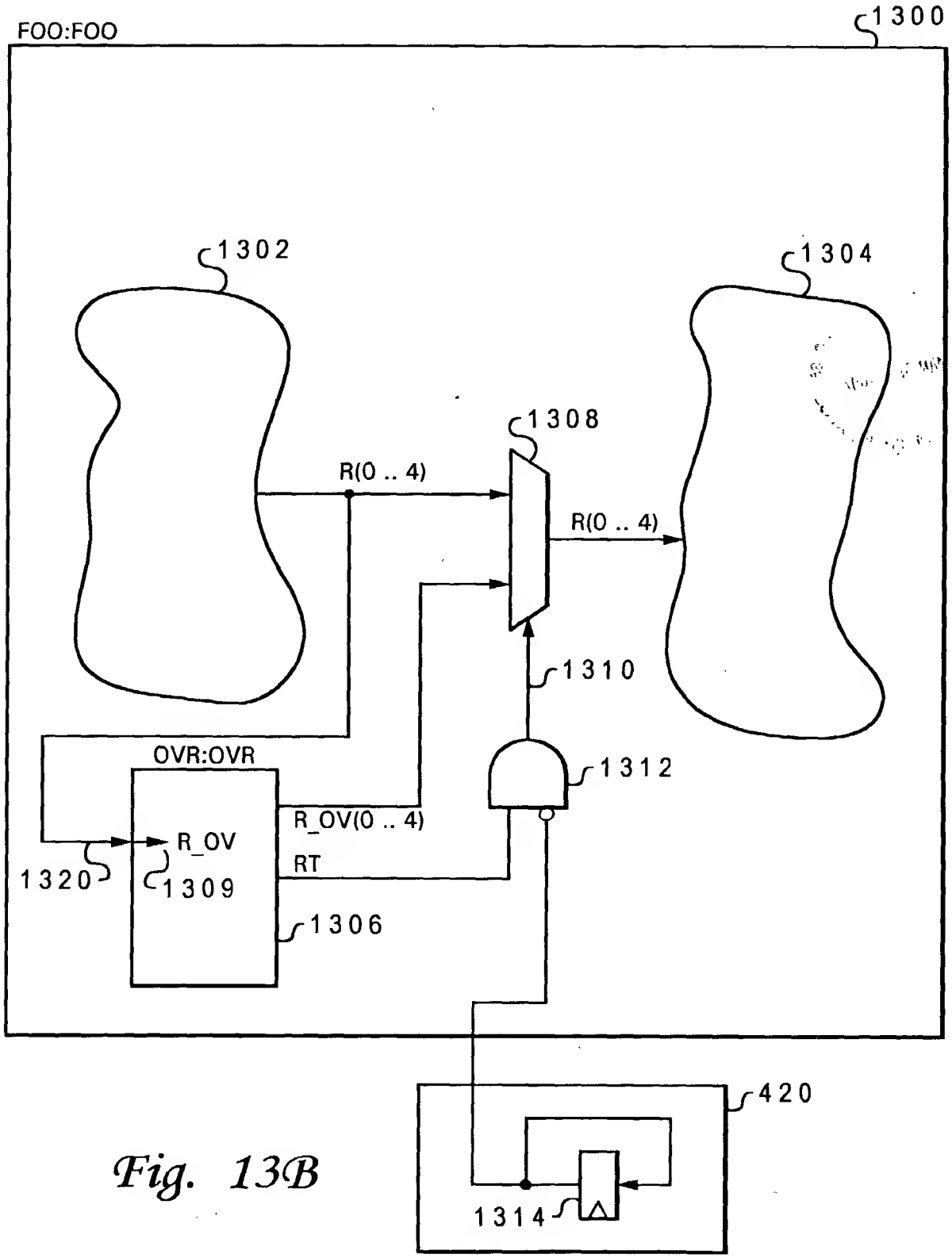

FOO:FOO
1300
1302
1304
1308
R(0 .. 4)
R(0 .. 4)
1310
OVR:OVR
1312
R_OV(0 .. 4)
R_OV
RT
1320
1309
1306
420
1314

Fig. 13B

```
ENTITY OVR IS                                          1364
    PORT(   R_IN        :    IN std_ulogic_vector(0 .. 4);
              .
              .
              .
            ... other ports as required ...
              .
              .                                        1362
              .
            R_OV        :    OUT std_ulogic_vector(0 .. 4);
            RT          :    OUT std_ulogic
         );                                            1363
--!! BEGIN
--!! Design Entity: FOO;

--!! Inputs (0 to 4)                                   1360
--!! R_IN => {R(0 .. 4)};
--!! :
... other ports as needed ...
--!! :
--!! End Inputs
                                                       1361
--!! Outputs
--!! <R_OVRRIDE> : R_OV(0 .. 4) => R(0 .. 4) [RT];
--!! End Outputs

--!! End

ARCHITECTURE example of OVR IS

BEGIN

    ... HDL code for entity body section ...

END;
```

1356 1351 1358 1340

Fig. 13C

```
ENTITY FOO IS

    PORT(    :
             :
             :
         );

ARCHITECTURE example of FOO IS

BEGIN

    .
    .
    .
    .
    .
    R <= ......
    .
    .
    .
    .
    .                    1381
    --!! R_IN <= {R};
    --!!                      1382
    --!!
    --!! R_OV(0 to 4) <= ......;        1383
    --!! RT <= ......;
    --!! [override, R_OVRRIDE, R(0 .. 4), RT] <= R_OV(0 to 4);
                                  1384
```

1380

Fig. 13D

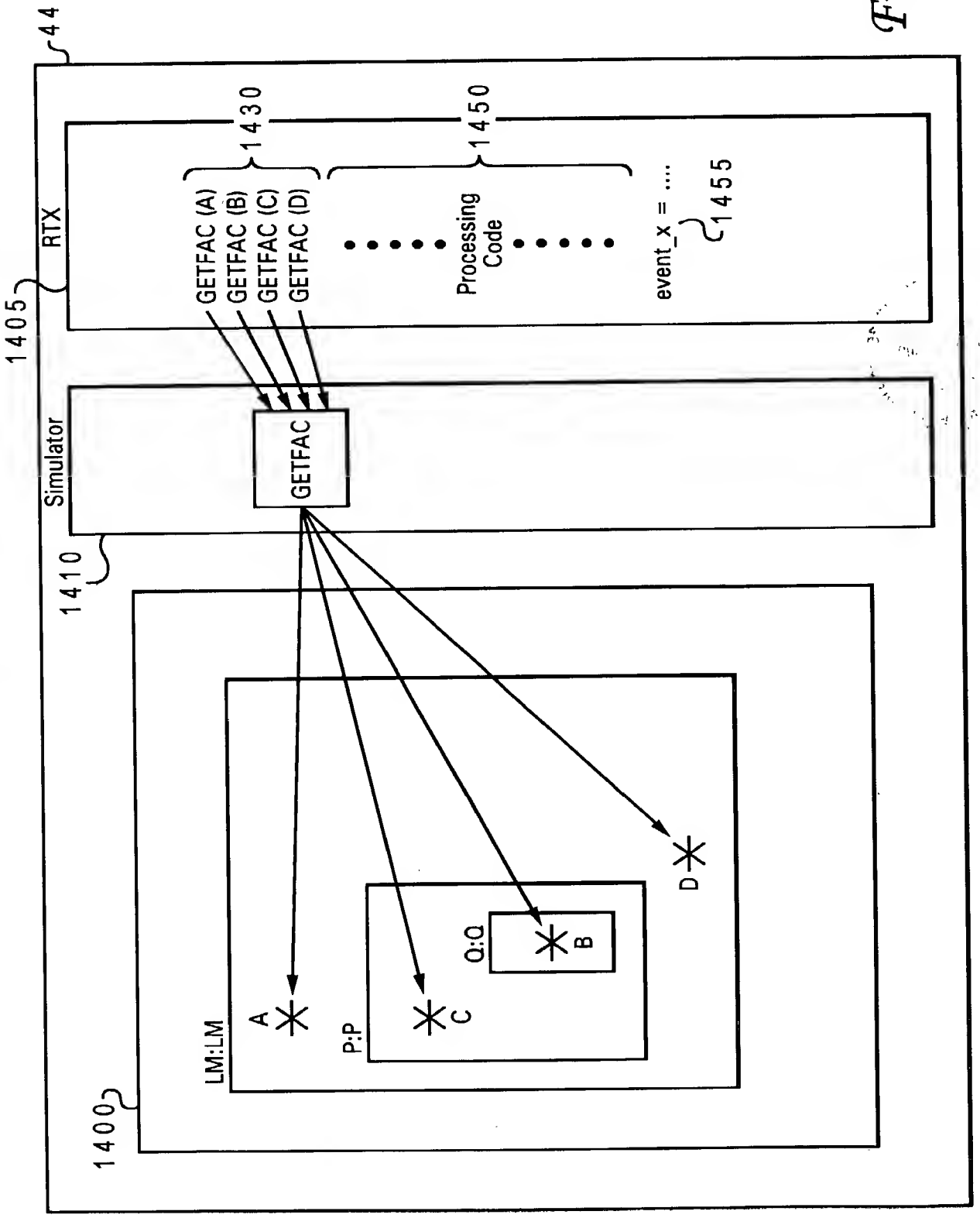

44
1405
RTX
1430
1450
GETFAC (A)
GETFAC (B)
GETFAC (C)
GETFAC (D)
Processing Code
event_x = ...
1455
Simulator
GETFAC
1410
1400
LM:LM
A
P:P
C
Q:Q
B
D

Fig. 14A

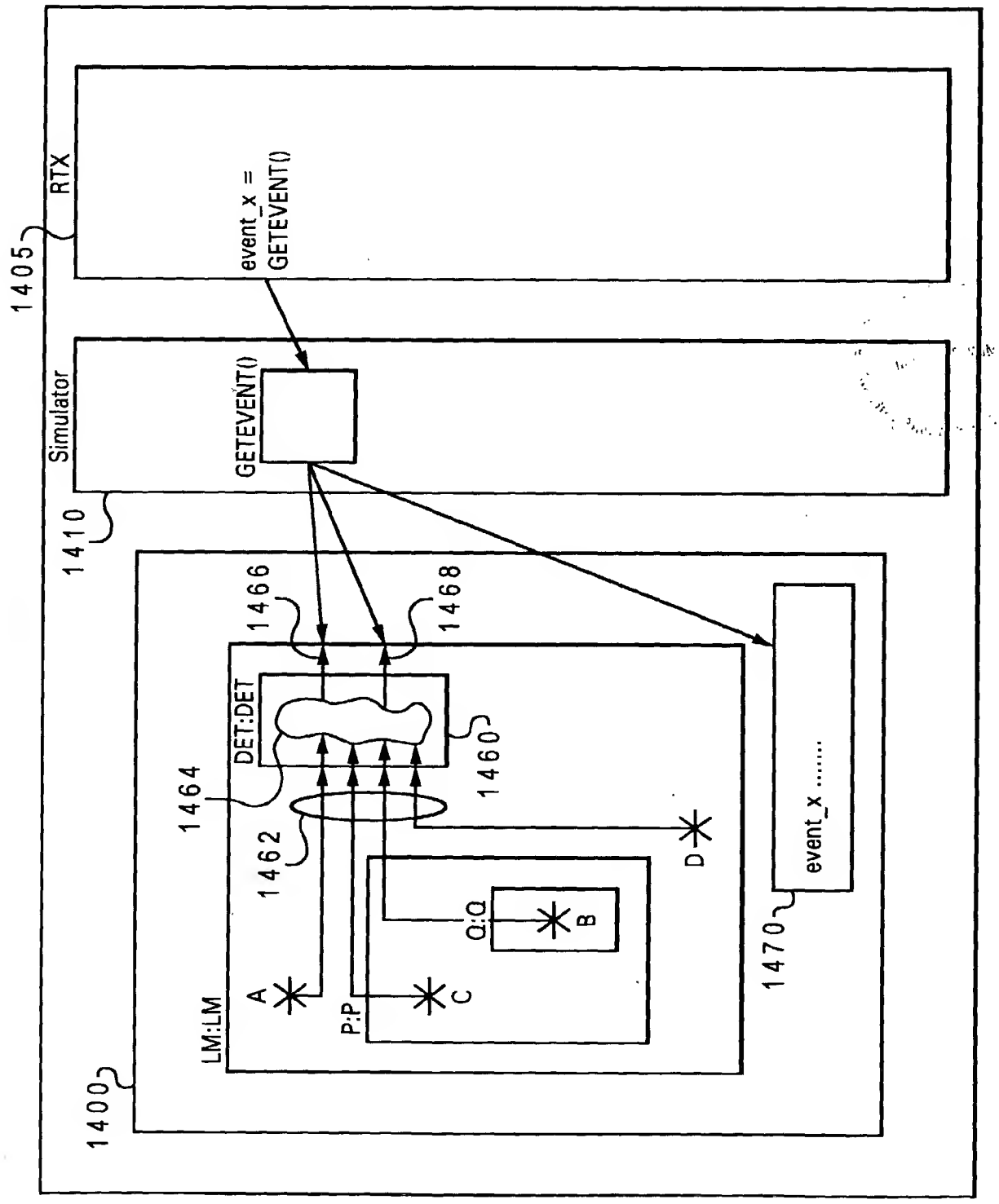

1405
RTX
event_x =
GETEVENT()
Simulator
GETEVENT()
1410
1466
1468
DET:DET
1464
1462
1460
1400
LM:LM
A
P:P
C
Q:Q
B
D
event_x .....
1470

Fig. 14B

```
ENTITY DET IS

    PORT(   A          :    IN std_ulogic;
            B          :    IN std_ulogic_vector(0 to 5);
            C          :    IN std_ulogic;
            D          :    IN std_ulogic;
                       :
                       :
            event_x    :    OUT std_ulogic_vector(0 to 2);
            x_here     :    OUT std_ulogic;
        );

--!! BEGIN
--!! Design Entity: LM;

--!! Inputs
--!! A     =>    A;
--!! B     =>    P.Q.B;
--!! C     =>    P.C;
--!! D     =>    D;
--!! End Inputs

--!! Detections
--!! <event_x>:event_x(0 to 2) [x_here];
--!! End Detections

--!! End;

ARCHITECTURE example of DET IS

BEGIN

    ... HDL code ...

END;
```

1491 1492 1493 1494 1495 1480

Fig. 14C

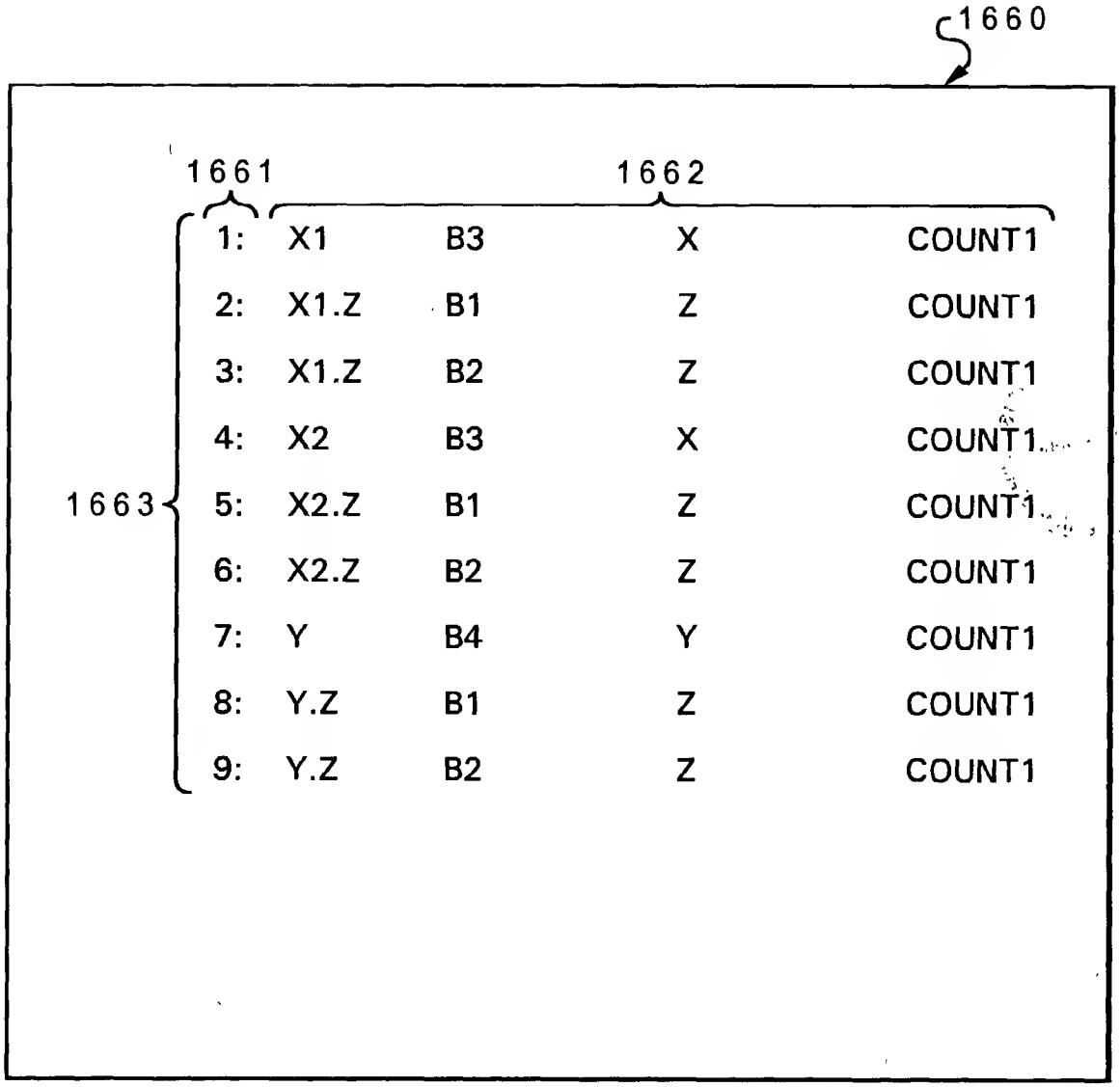

1660
1661
1662
1663
1: X1 B3 X COUNT1
2: X1.Z B1 Z COUNT1
3: X1.Z B2 Z COUNT1
4: X2 B3 X COUNT1
5: X2.Z B1 Z COUNT1
6: X2.Z B2 Z COUNT1
7: Y B4 Y COUNT1
8: Y.Z B1 Z COUNT1
9: Y.Z B2 Z COUNT1

Fig. 15

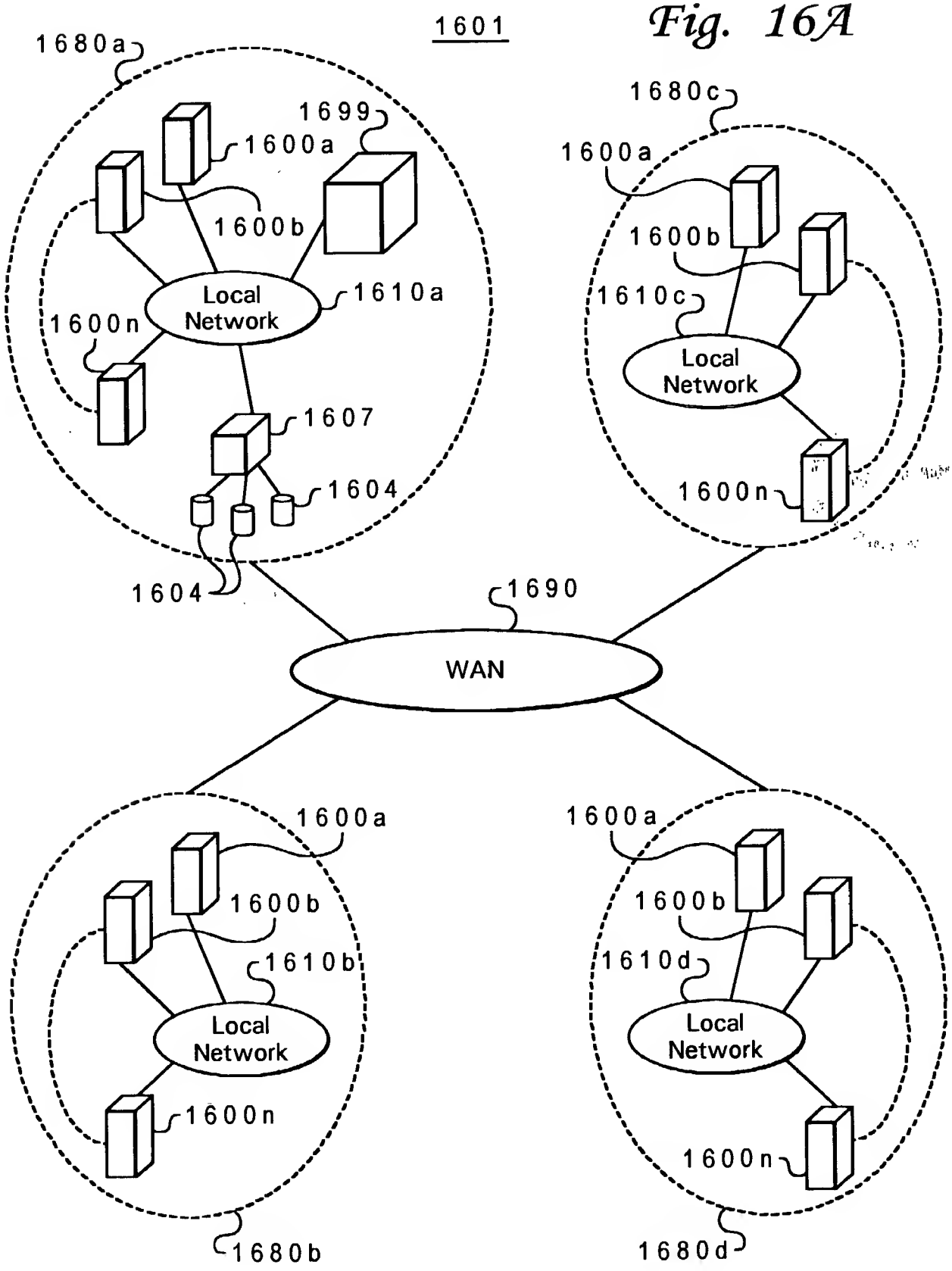

1601
1680a
1699
1600a
1600b
1610a
Local Network
1600n
1607
1604
1604
1680c
1600a
1600b
1610c
Local Network
1600n
1690
WAN
1600a
1600b
1610b
Local Network
1600n
1680b
1600a
1600b
1610d
Local Network
1600n
1680d

Fig. 16A

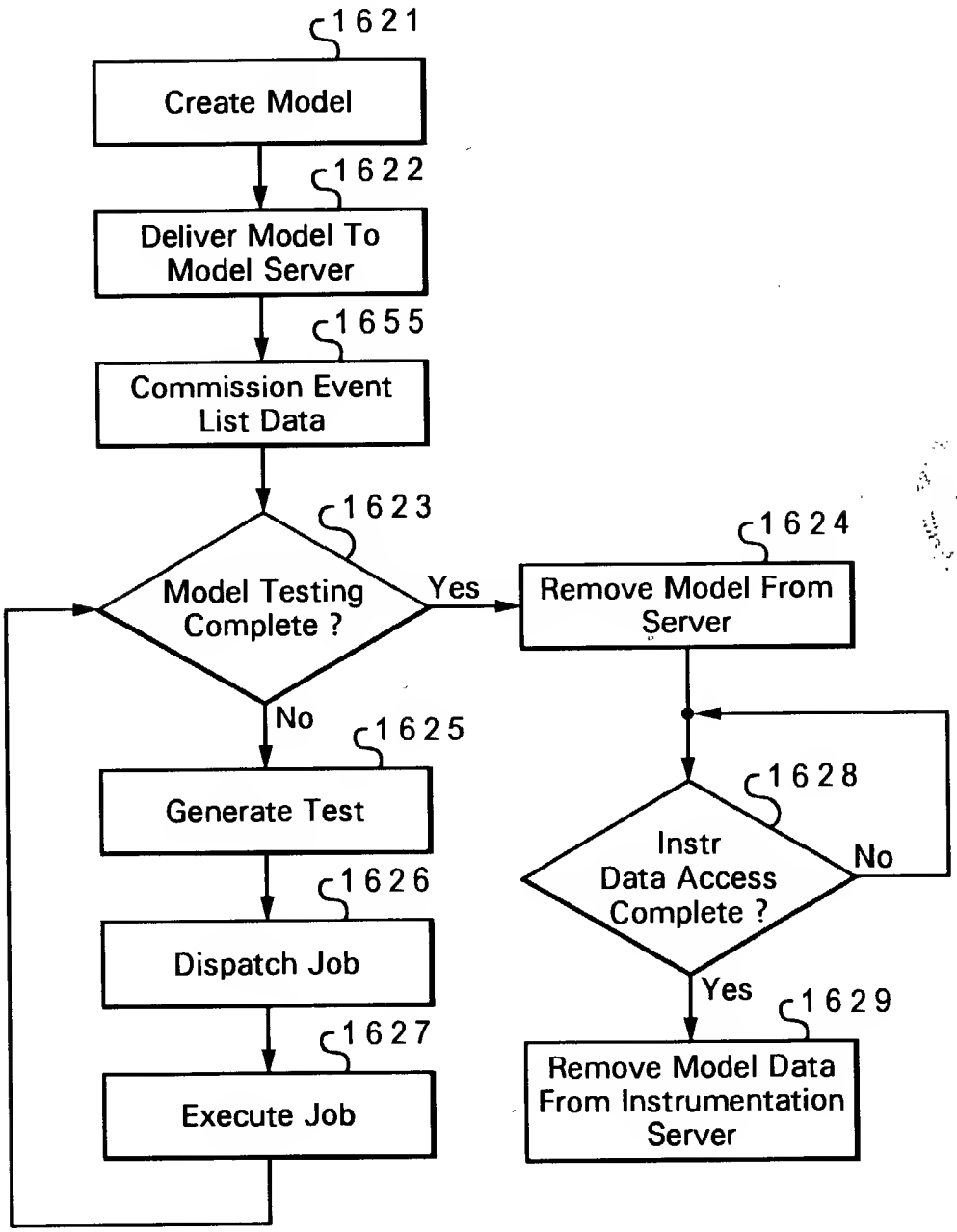

1621
Create Model
1622
Deliver Model To Model Server
1655
Commission Event List Data
1623
Model Testing Complete ?
Yes
No
1624
Remove Model From Server
1625
Generate Test
1626
Dispatch Job
1627
Execute Job
1628
Instr Data Access Complete ?
No
Yes
1629
Remove Model Data From Instrumentation Server

Fig. 16B

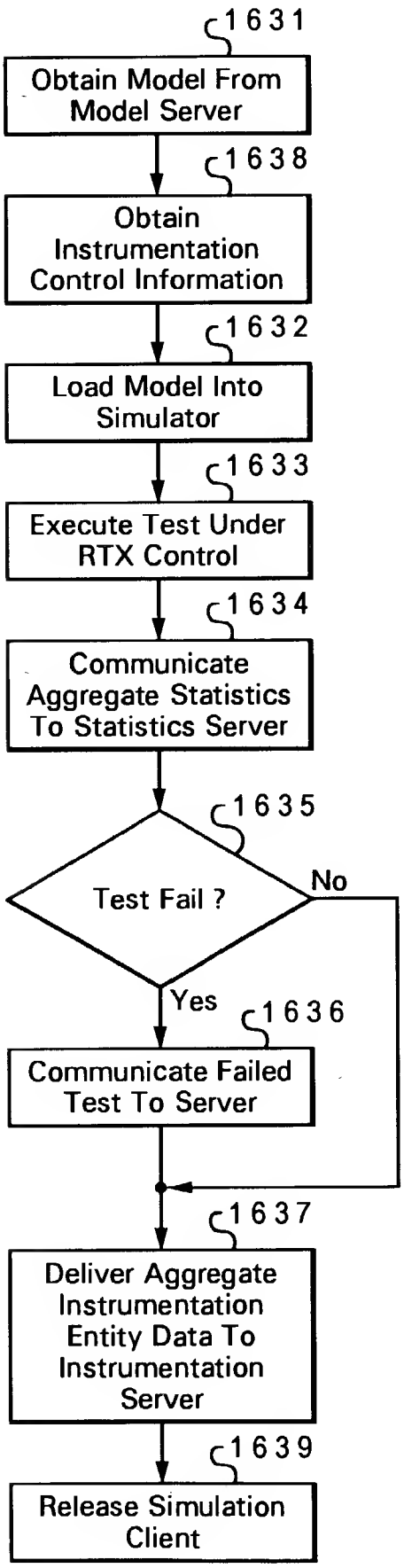

1631
Obtain Model From Model Server
1638
Obtain Instrumentation Control Information
1632
Load Model Into Simulator
1633
Execute Test Under RTX Control
1634
Communicate Aggregate Statistics To Statistics Server
1635
Test Fail ?
No
Yes
1636
Communicate Failed Test To Server
1637
Deliver Aggregate Instrumentation Entity Data To Instrumentation Server
1639
Release Simulation Client

Fig. 16C

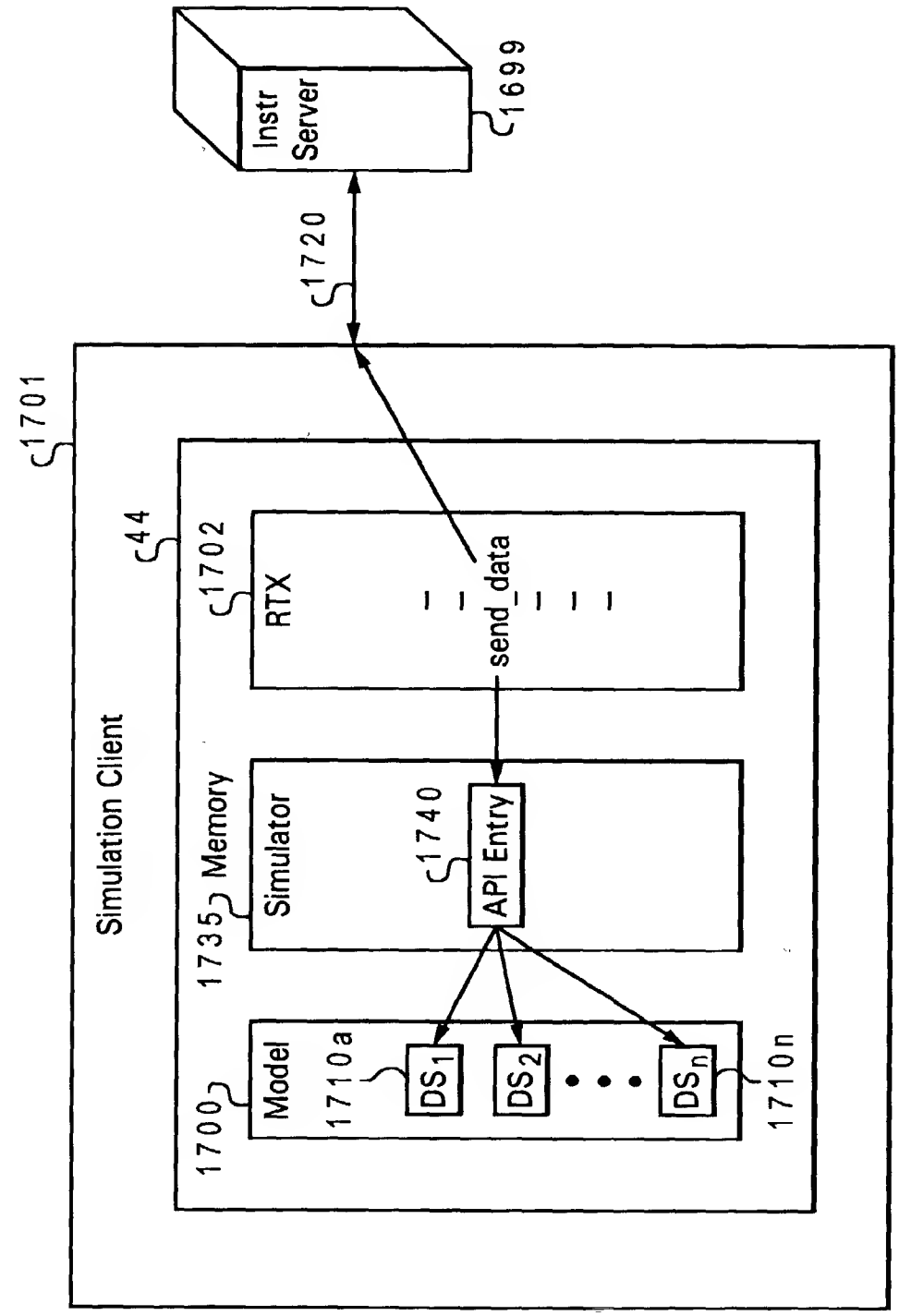

Instr Server
1699
1720
1701
Simulation Client
44
1702
RTX
send_data
1735 Memory
Simulator
1740
API Entry
1700
Model
1710a
DS1
DS2
DSn
1710n

Fig. 17A

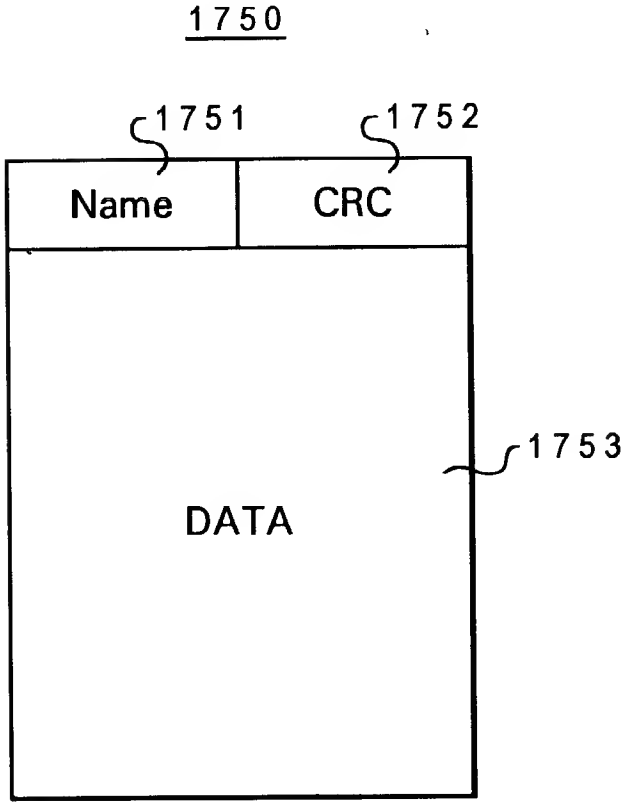


Fig. 17B

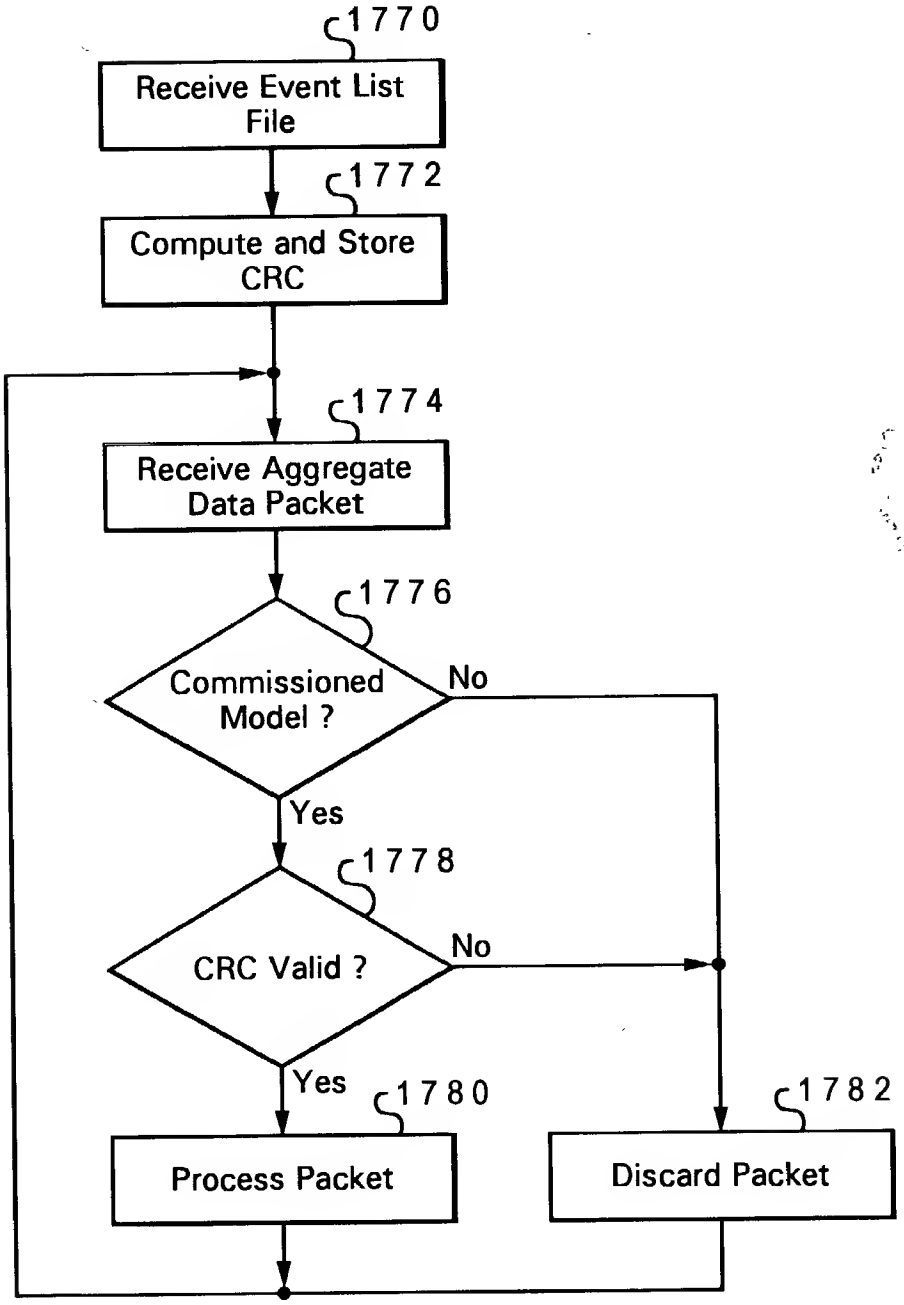

1770
Receive Event List File
1772
Compute and Store CRC
1774
Receive Aggregate Data Packet
1776
Commissioned Model ?
No
Yes
1778
CRC Valid ?
No
Yes
1780
Process Packet
1782
Discard Packet

Fig. 17C

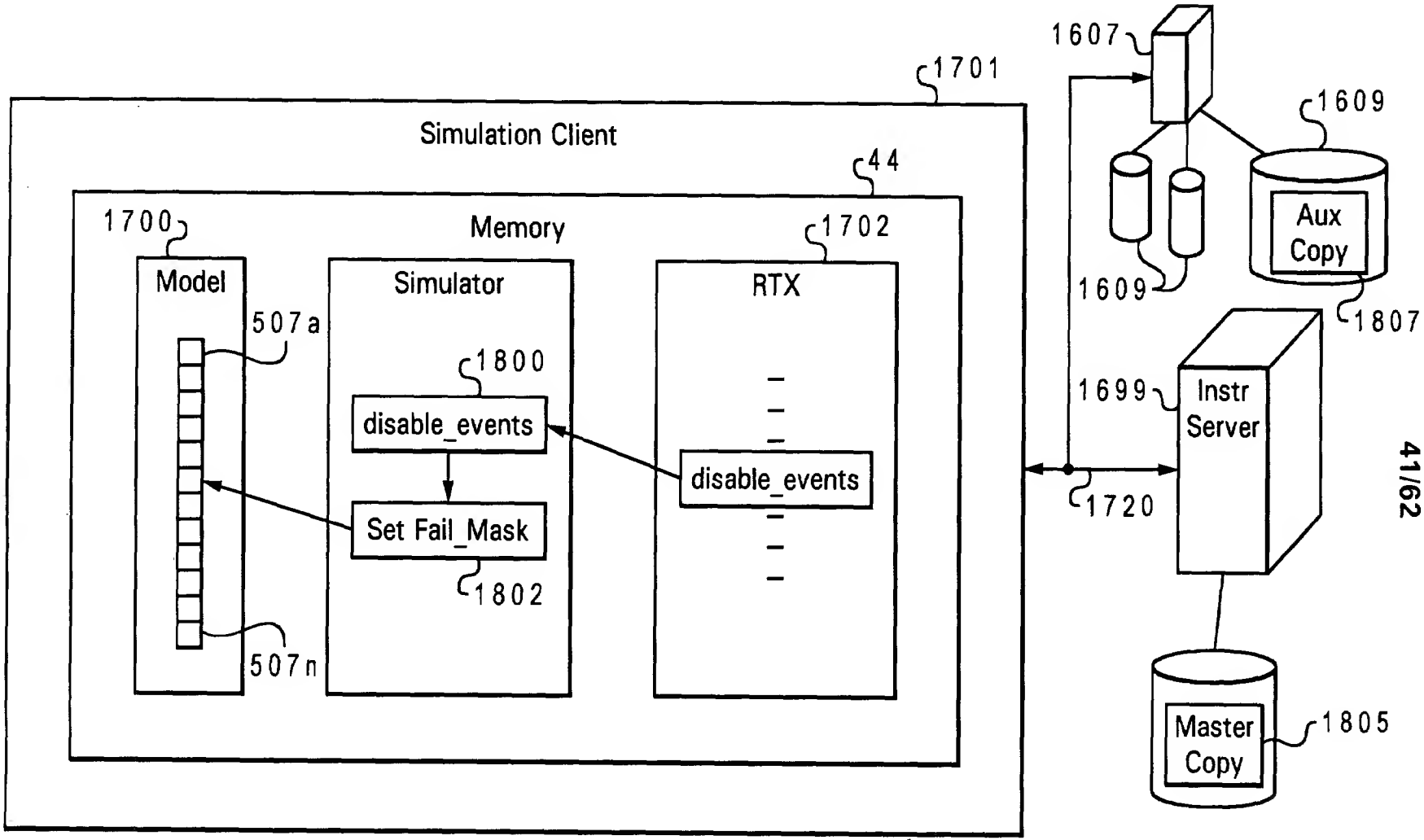

1607
1701
1609
Simulation Client
44
Aux
Copy
1700
Memory
1702
Model
Simulator
RTX
507a
1800
1609
1807
1699
Instr
Server
disable_events
disable_events
1720
Set Fail_Mask
1802
507n
Master
Copy
1805

Fig. 18A

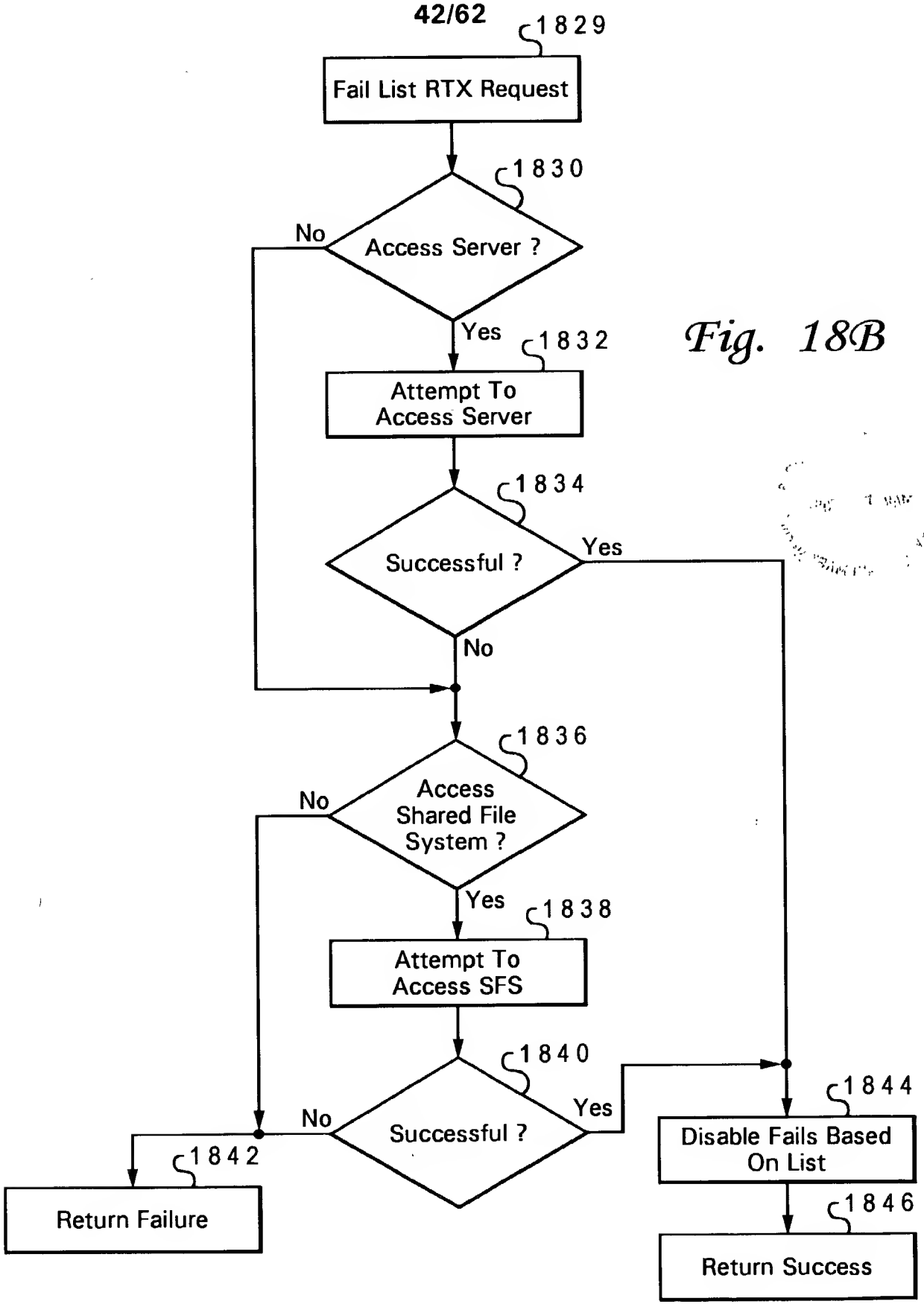

1829
Fail List RTX Request
1830
Access Server ?
No
Yes
1832
Attempt To Access Server
1834
Successful ?
Yes
No
1836
Access Shared File System ?
No
Yes
1838
Attempt To Access SFS
1840
Successful ?
No
Yes
1842
Return Failure
1844
Disable Fails Based On List
1846
Return Success

Fig. 18B

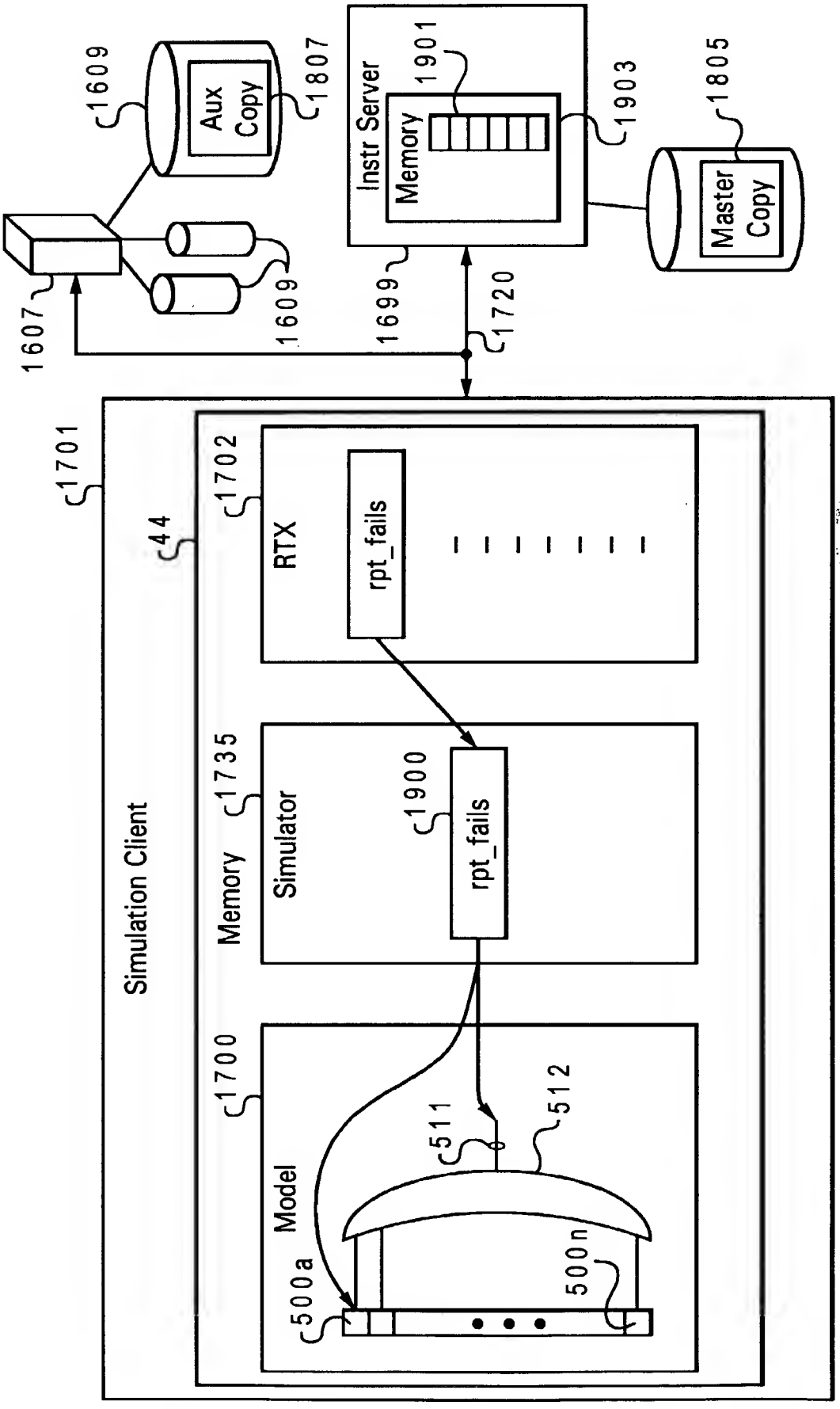

1609
Aux Copy
1807
Instr Server
1901
Memory
1903
1805
Master Copy
1607
1609
1699
1720
1701
44
1702
RTX
rpt_fails
Simulation Client
Memory
1735
Simulator
1900
rpt_fails
1700
Model
511
512
500a
500n

Fig. 19A

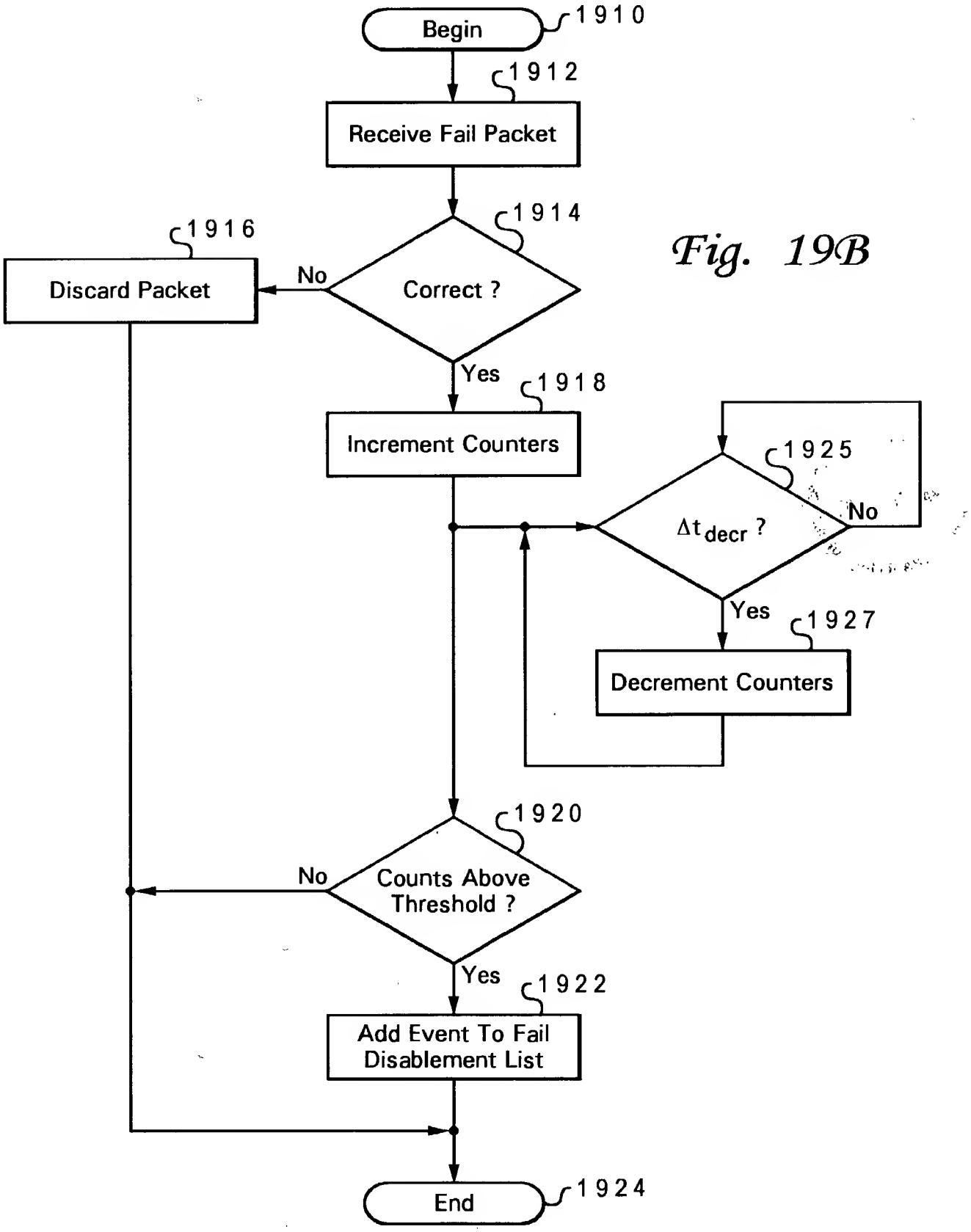

Begin
1910
1912
Receive Fail Packet
1914
Correct ?
No
1916
Discard Packet
Yes
1918
Increment Counters
1925
Δt_{decr} ?
No
Yes
1927
Decrement Counters
1920
Counts Above Threshold ?
No
Yes
1922
Add Event To Fail Disablement List
End
1924

Fig. 19B

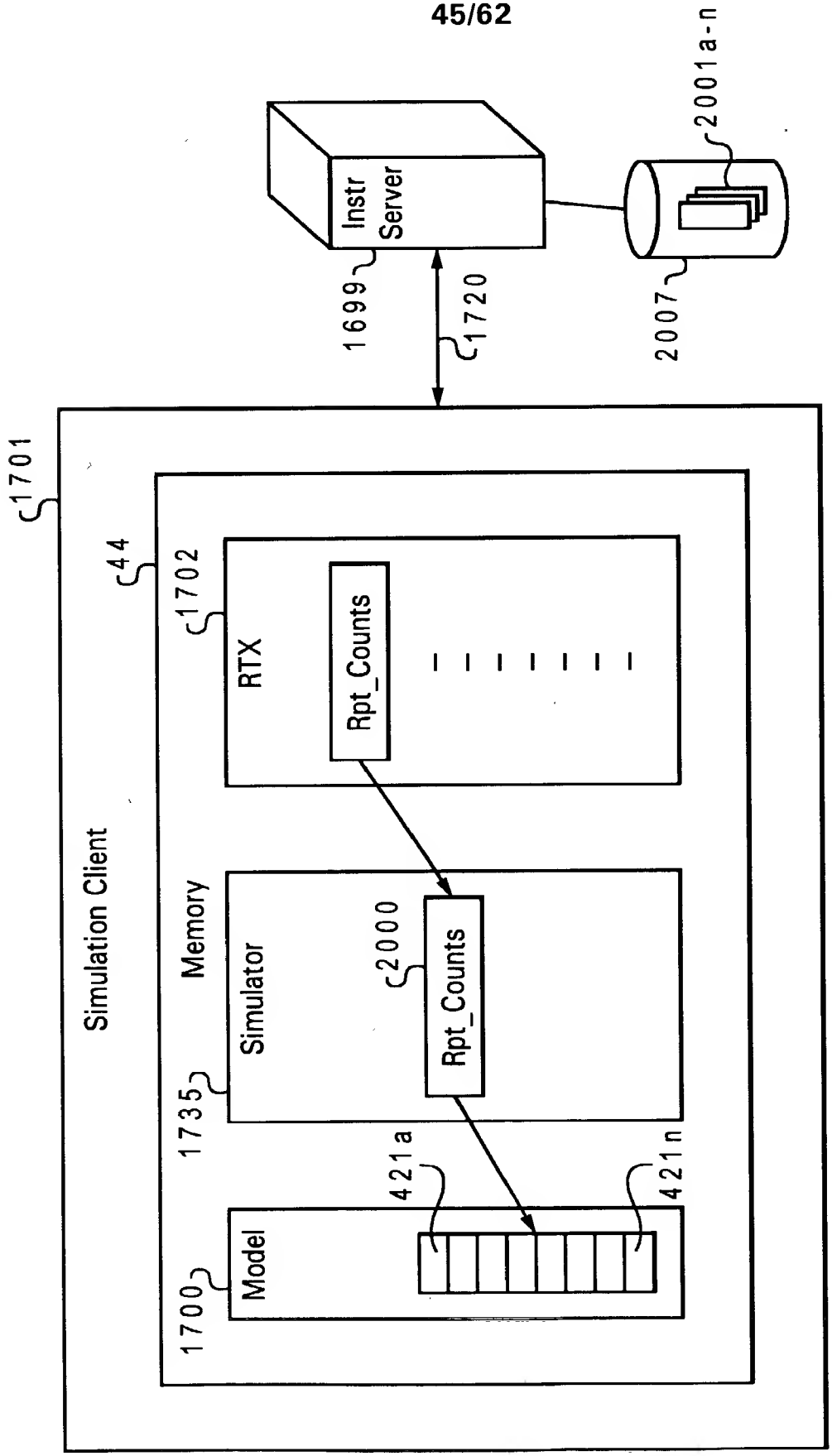

2001a-n
Instr Server
1699
1720
2007
1701
Simulation Client
44
Memory
1702
RTX
Rpt_Counts
Simulator
2000
Rpt_Counts
1735
421a
421n
Model
1700

Fig. 20A

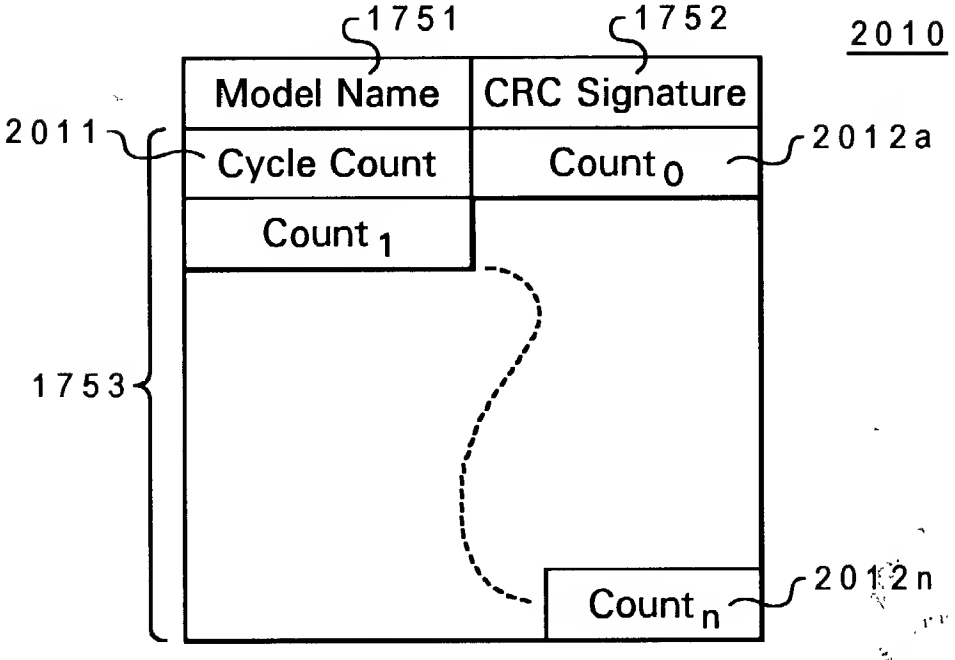


Fig. 20B

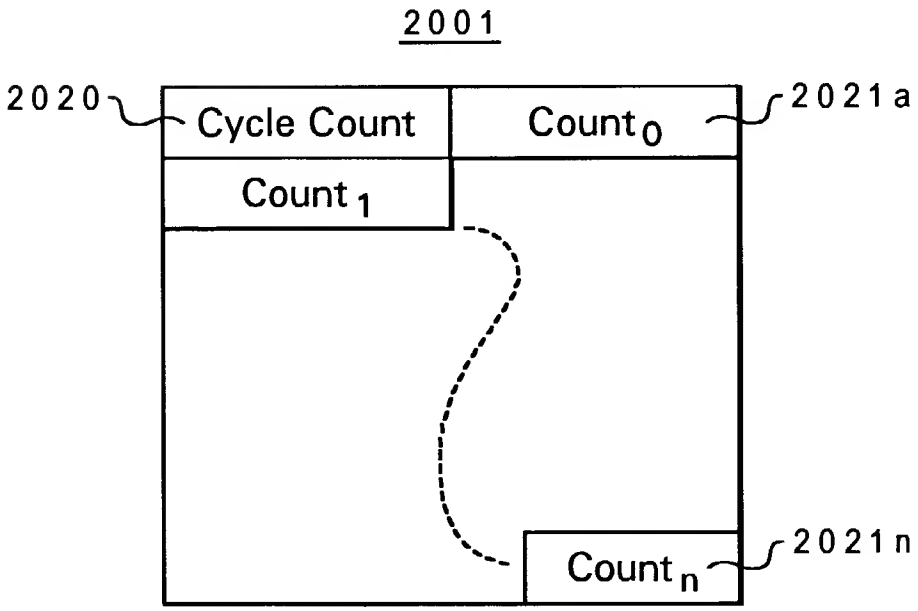


Fig. 20C

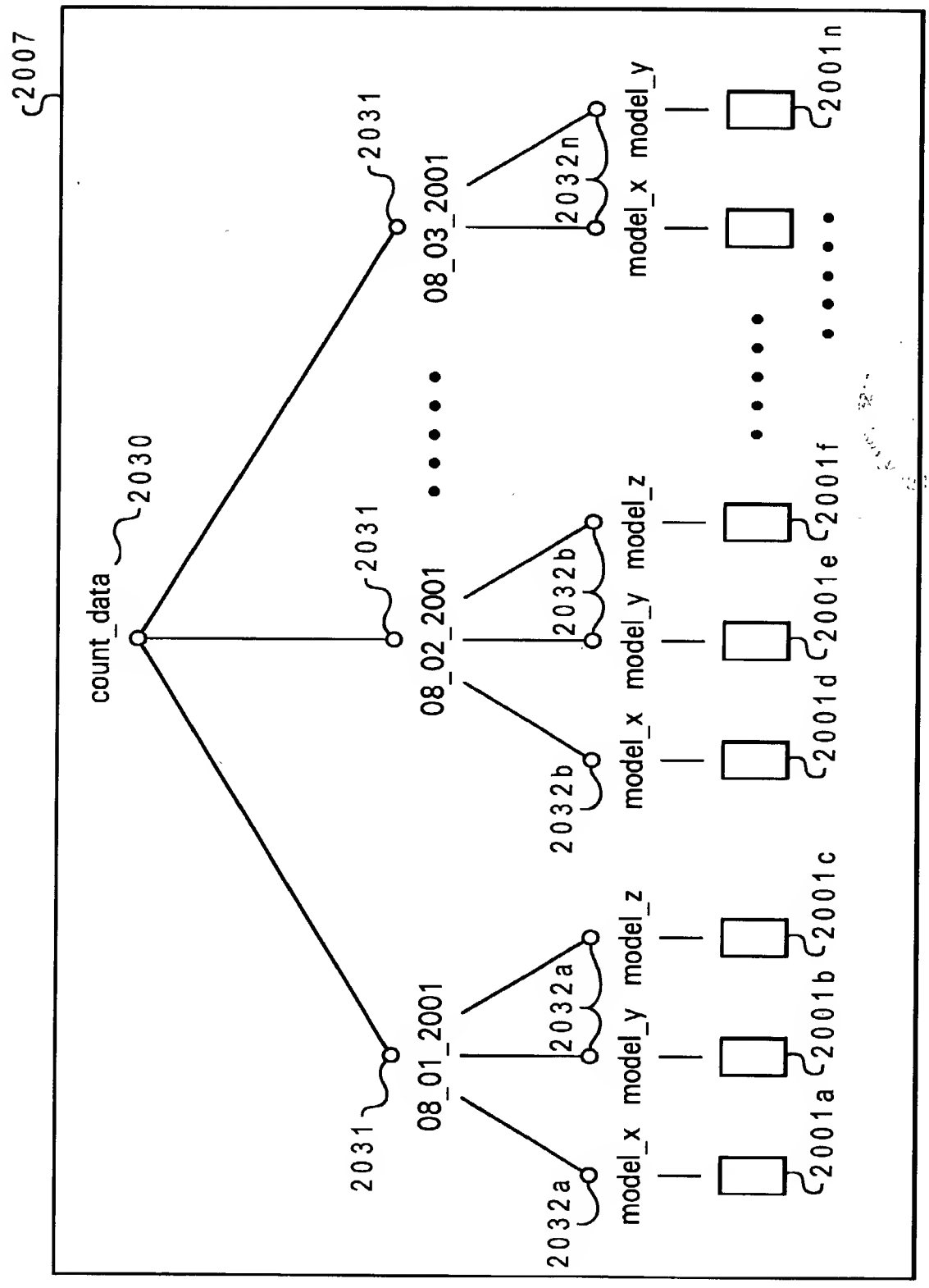

2007
count_data
2030
2031
08_01_2001
2032a
model_x
model_y
model_z
2001a
2001b
2001c
2031
08_02_2001
2032b
2001d
2001e
2001f
2031
08_03_2001
2032n
2001n

Fig. 20D

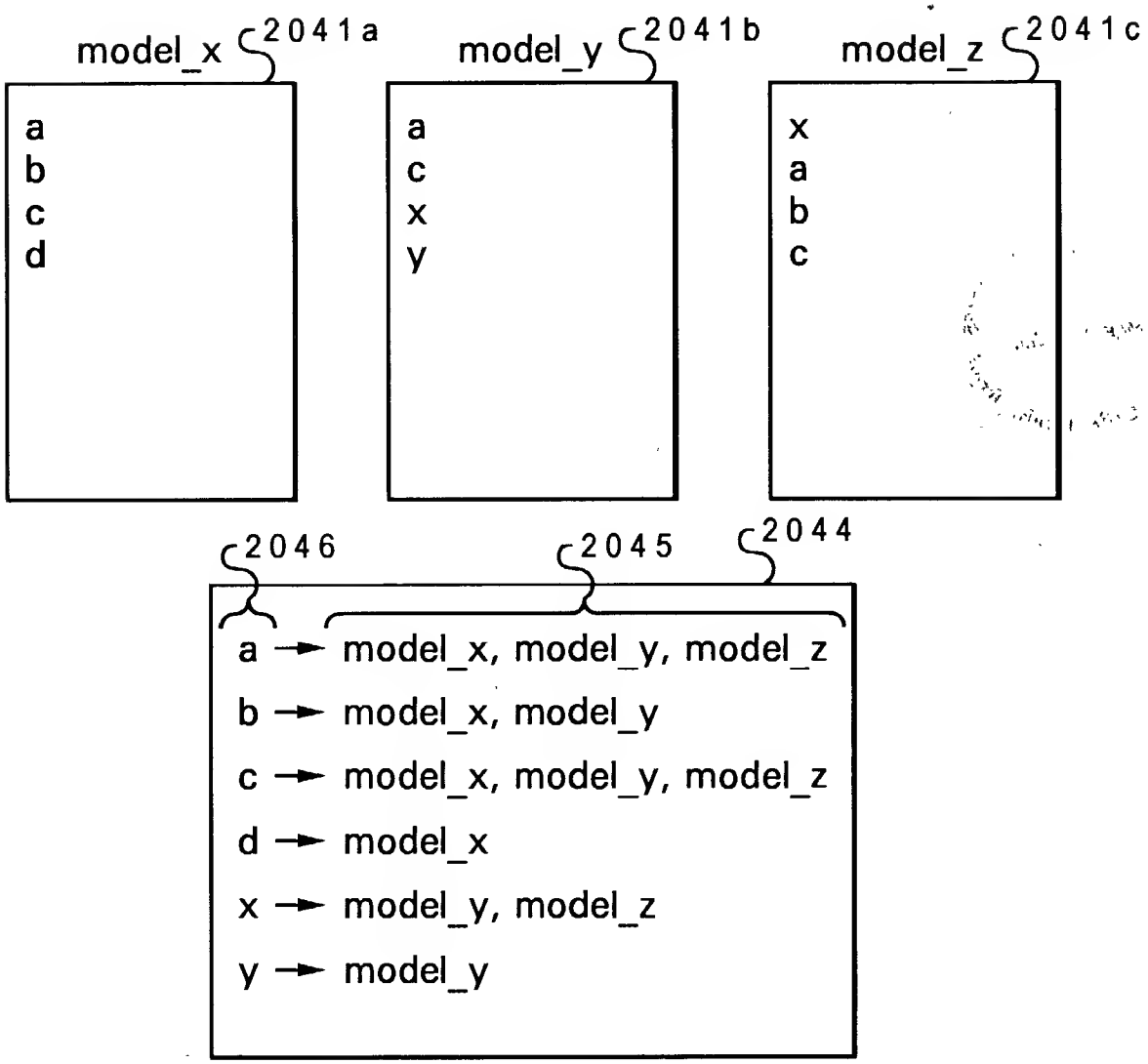

model_x
2041a
a
b
c
d
model_y
2041b
a
c
x
y
model_z
2041c
x
a
b
c
2046
2045
2044
a → model_x, model_y, model_z
b → model_x, model_y
c → model_x, model_y, model_z
d → model_x
x → model_y, model_z
y → model_y

Fig. 20E

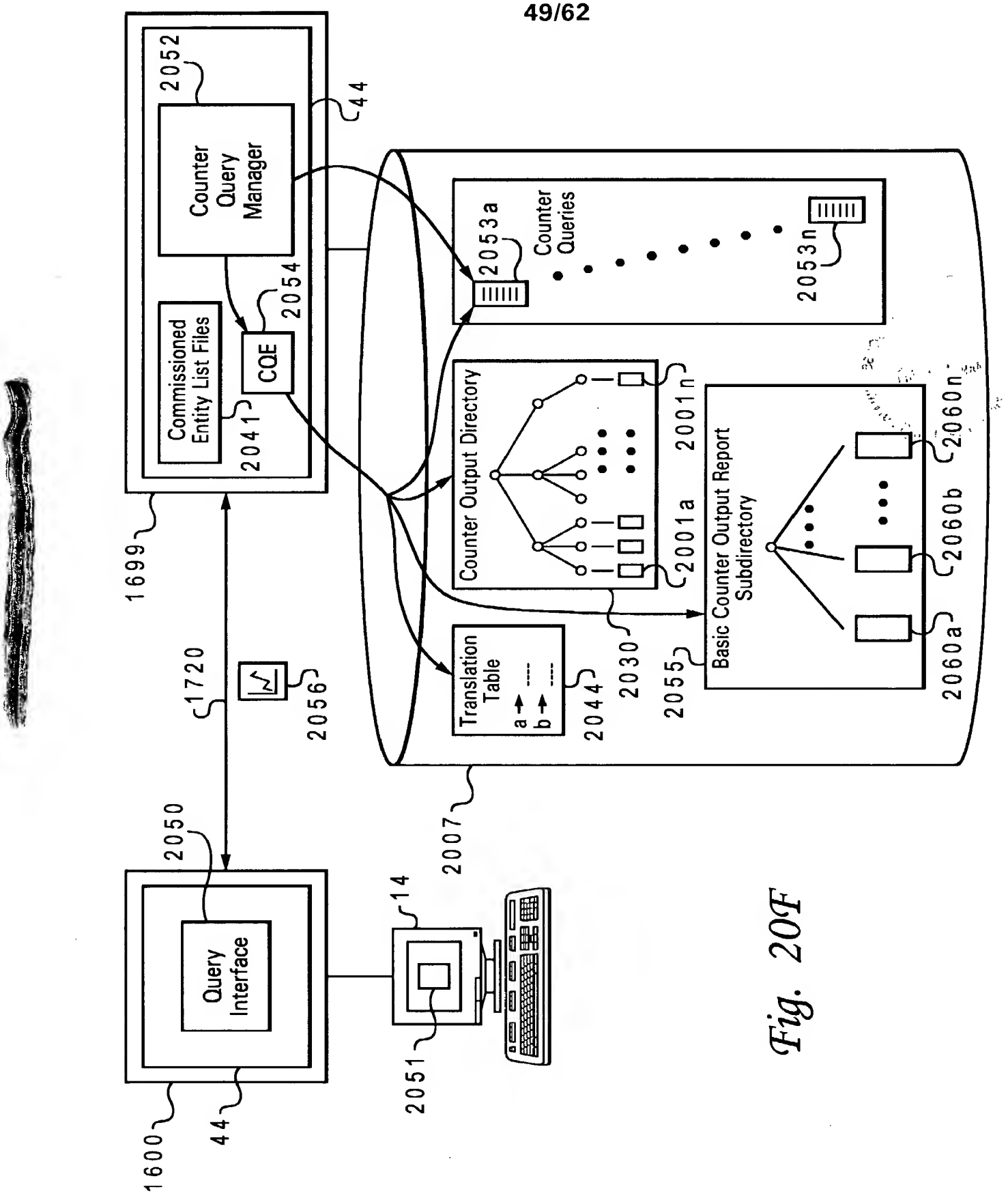

1600
44
Query Interface
2050
1720
1699
2052
Counter Query Manager
44
Commissioned Entity List Files
2041
COE
2054
2056
2007
14
2051
Translation Table
a
b
2044
2030
2055
Counter Output Directory
2001a
2001n
Basic Counter Output Report Subdirectory
2060a
2060b
2060n
Counter Queries
2053a
2053n

Fig. 20F

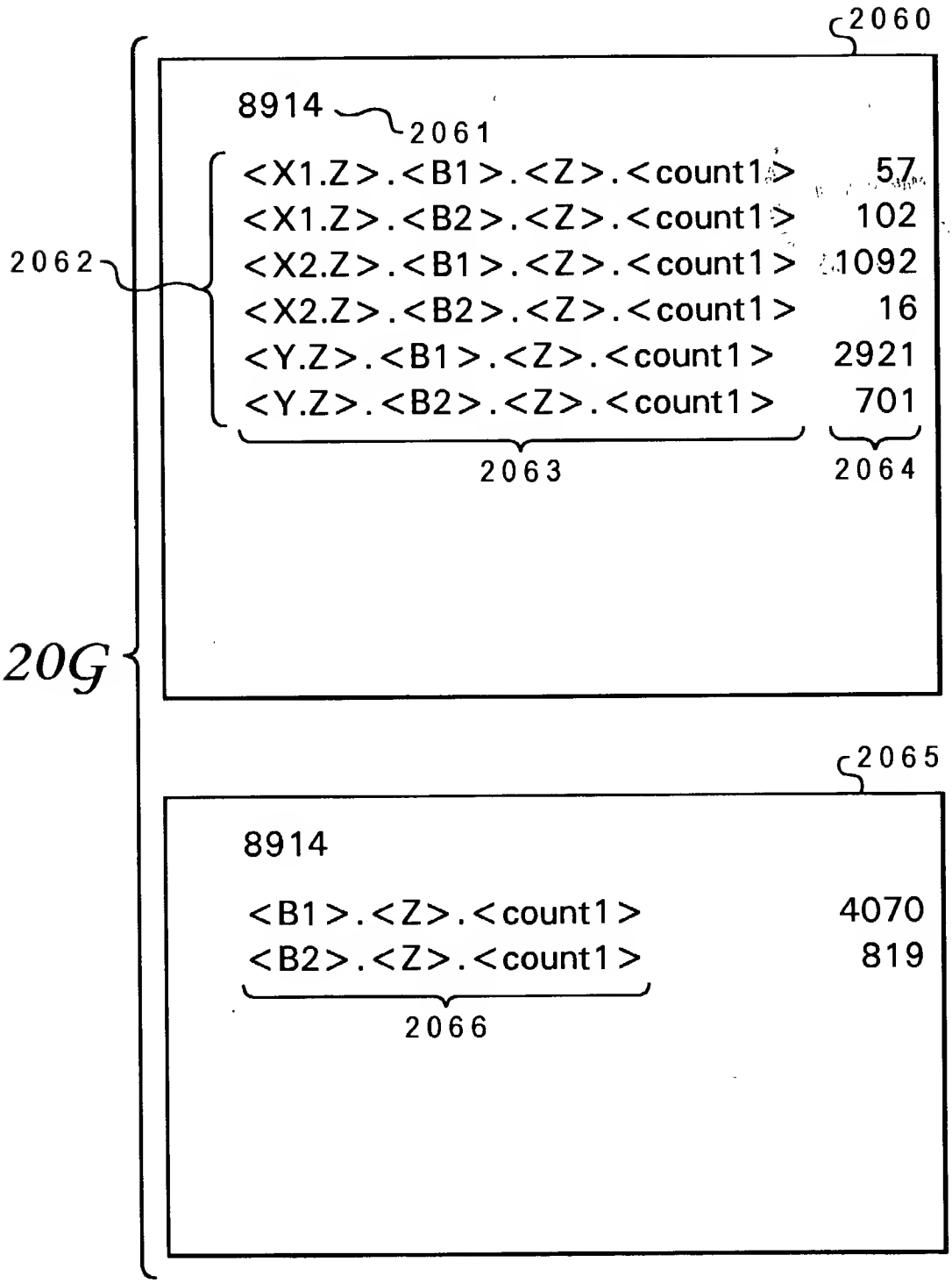

2060
8914
2061
<X1.Z>.<B1>.<Z>.<count1> 57
<X1.Z>.<B2>.<Z>.<count1> 102
2062
<X2.Z>.<B1>.<Z>.<count1> 1092
<X2.Z>.<B2>.<Z>.<count1> 16
<Y.Z>.<B1>.<Z>.<count1> 2921
<Y.Z>.<B2>.<Z>.<count1> 701
2063
2064
2065
8914
<B1>.<Z>.<count1> 4070
<B2>.<Z>.<count1> 819
2066

Fig. 20G

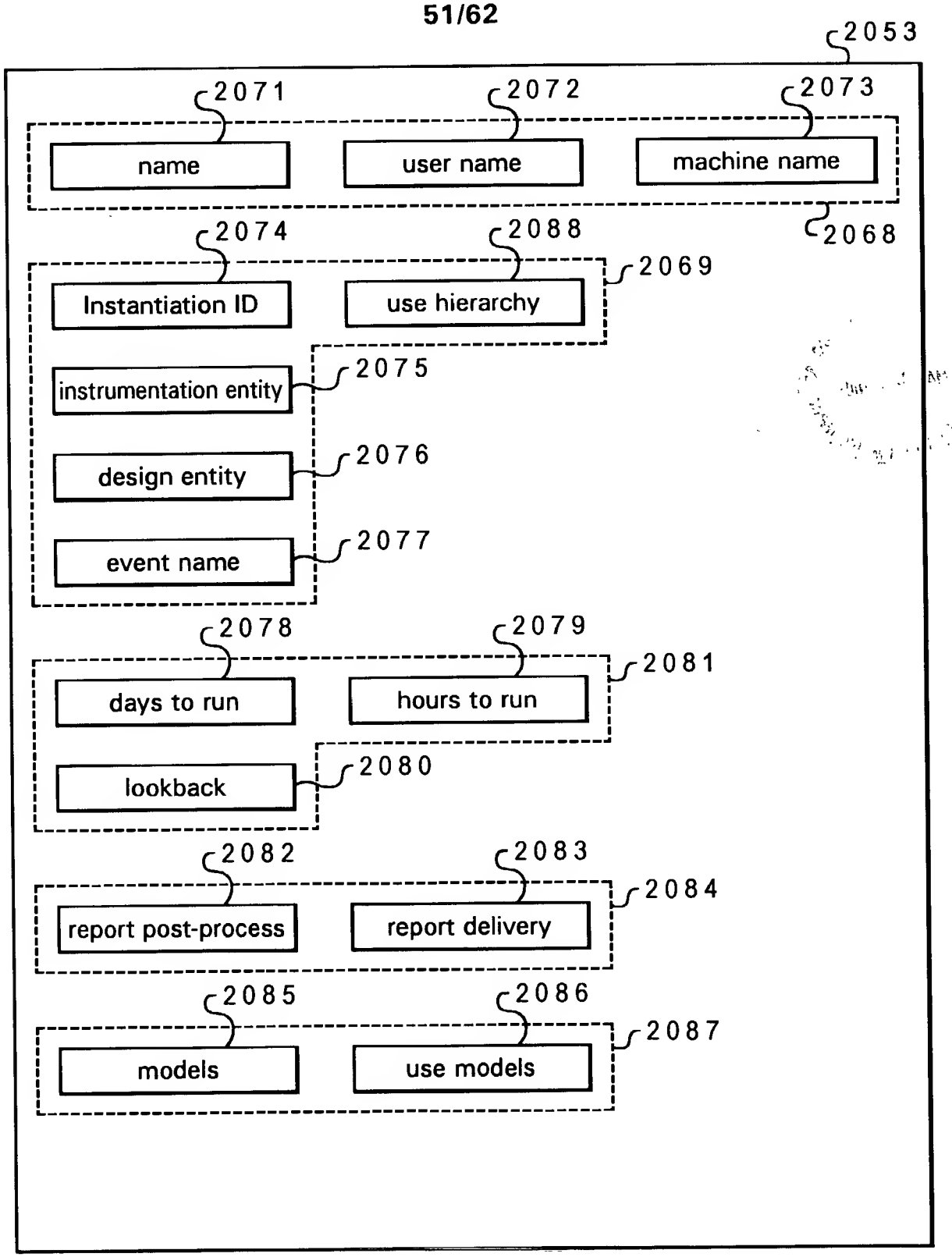

2053
2071
2072
2073
name
user name
machine name
2068
2074
2088
2069
Instantiation ID
use hierarchy
instrumentation entity
2075
design entity
2076
event name
2077
2078
2079
2081
days to run
hours to run
lookback
2080
2082
2083
2084
report post-process
report delivery
2085
2086
2087
models
use models

Fig. 20H

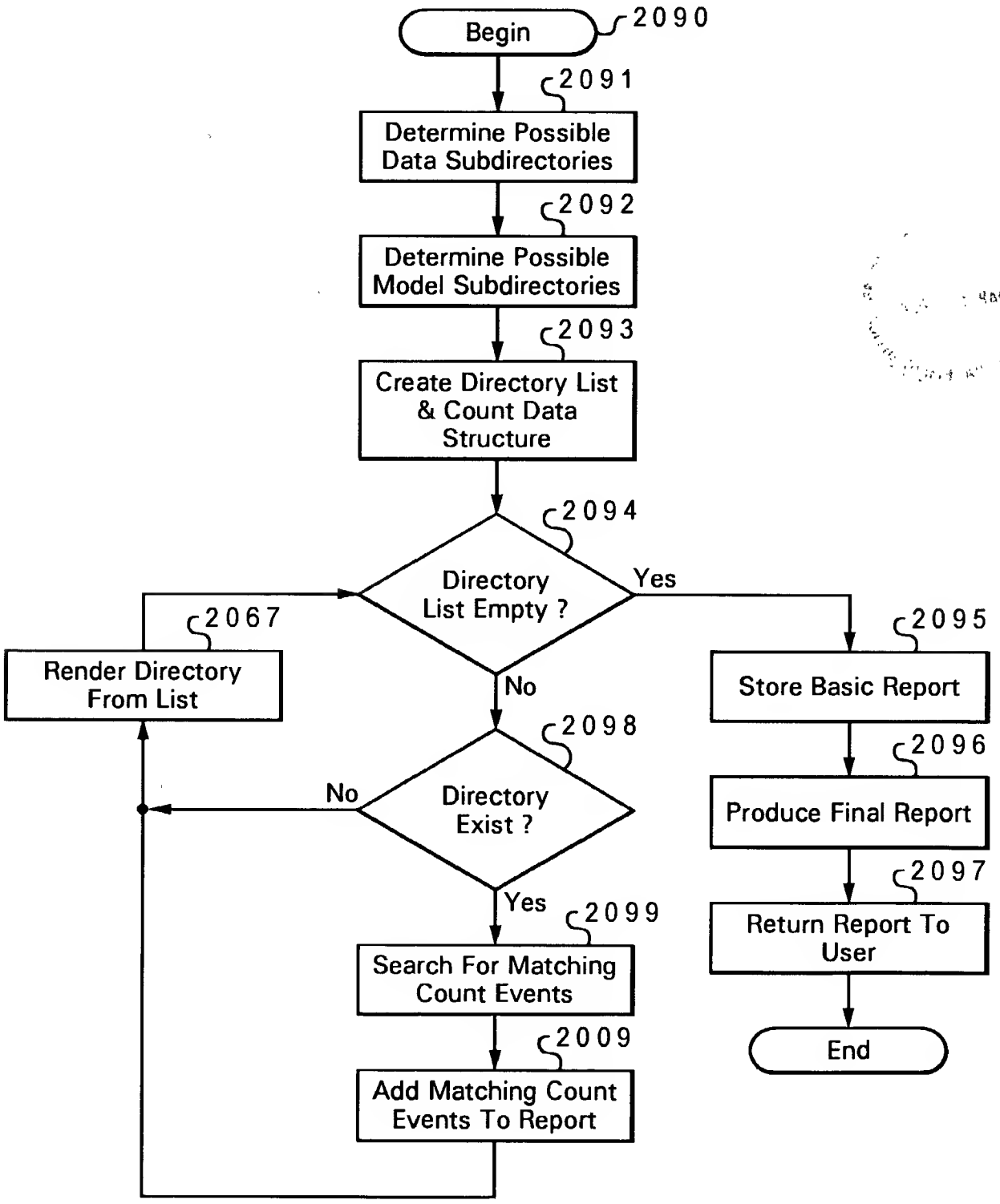

Begin
2090
2091
Determine Possible Data Subdirectories
2092
Determine Possible Model Subdirectories
2093
Create Directory List & Count Data Structure
2094
Directory List Empty ?
Yes
No
2067
Render Directory From List
2095
Store Basic Report
2096
Produce Final Report
2097
Return Report To User
End
2098
Directory Exist ?
No
Yes
2099
Search For Matching Count Events
2009
Add Matching Count Events To Report

Fig. 20I

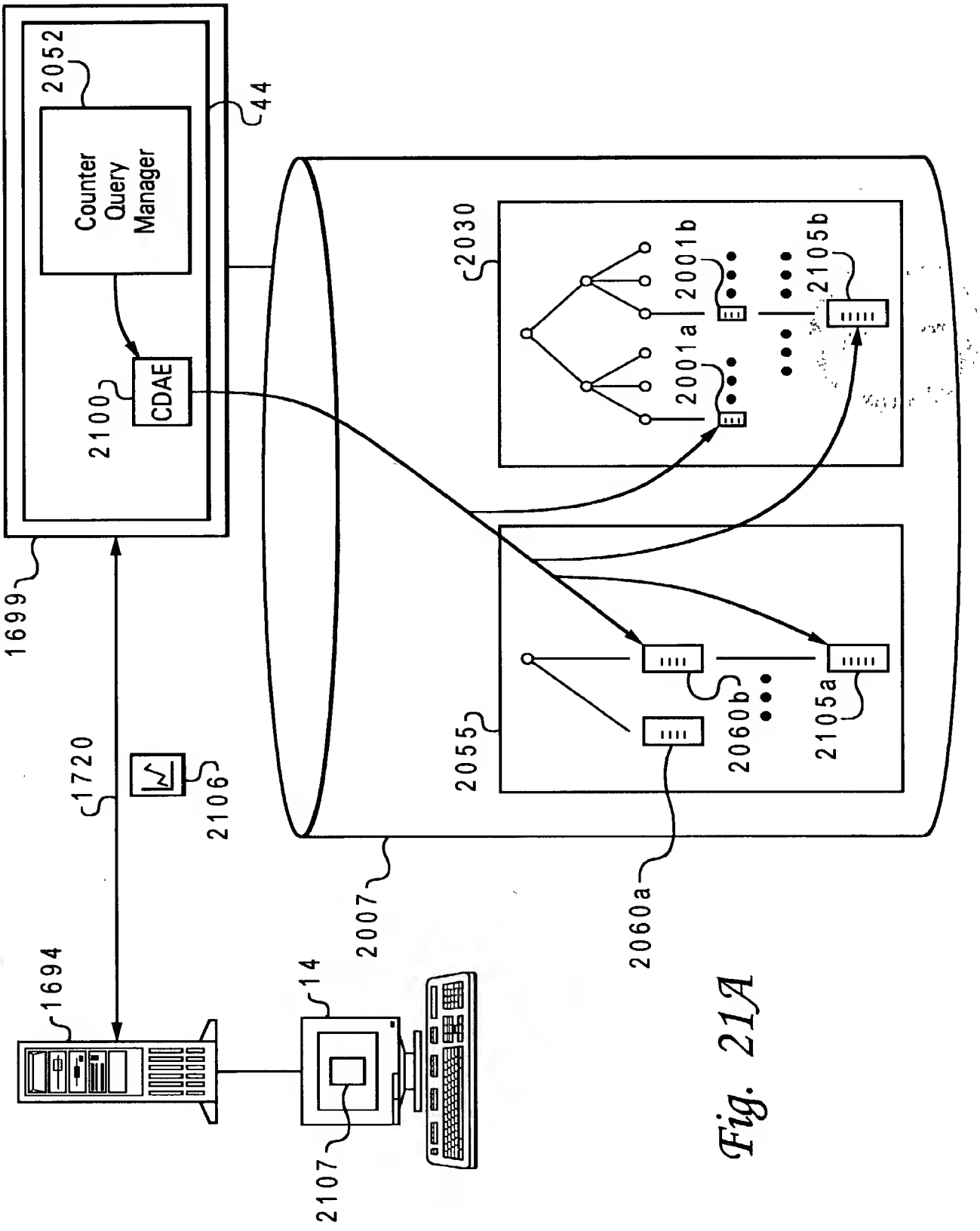

2052
Counter Query Manager
44
2100
CDAE
1699
1720
2106
1694
14
2107
2007
2030
2001a
2001b
2105b
2055
2060b
2105a
2060a

Fig. 21A

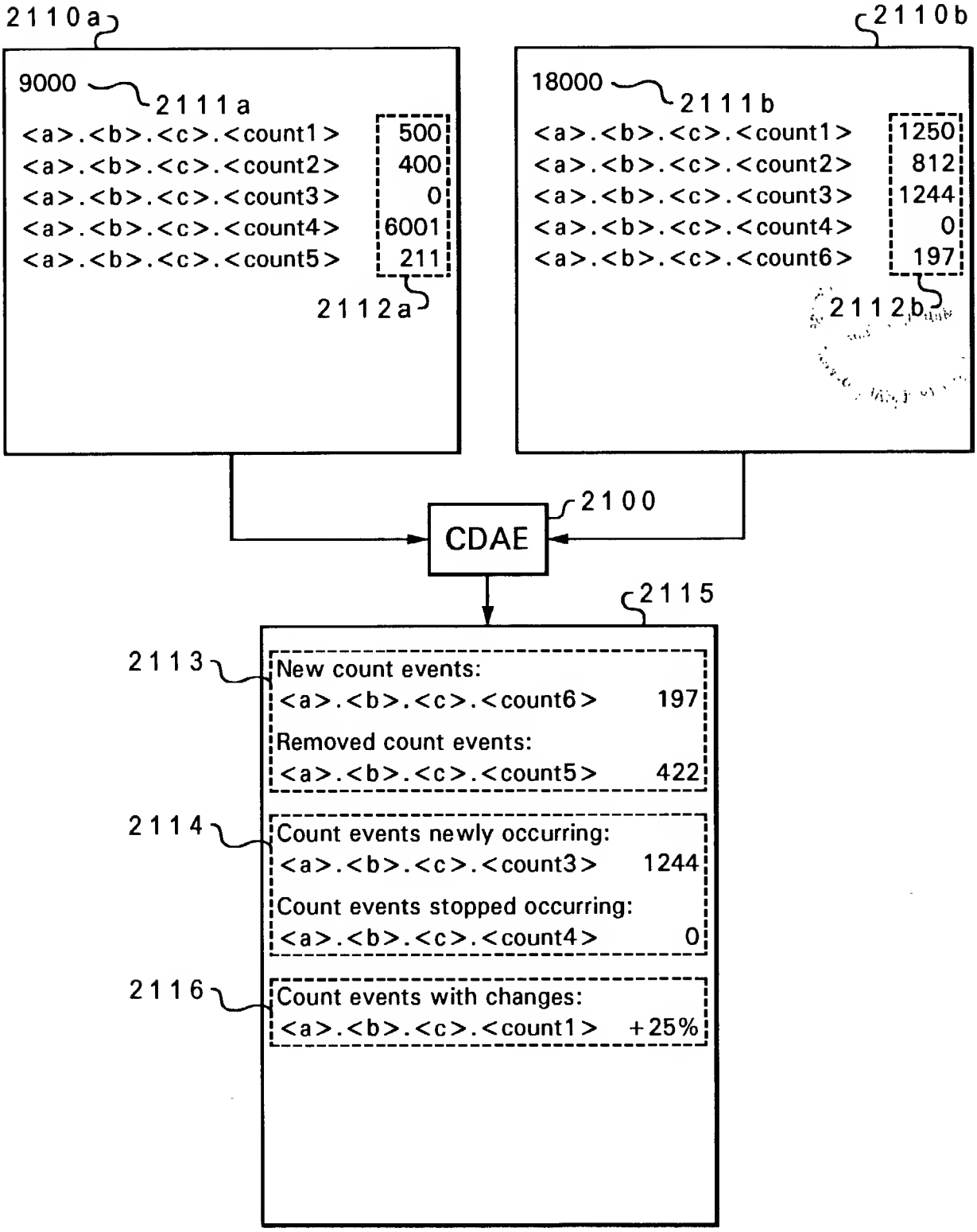

2110a
9000
2111a
<a>.<b>.<c>.<count1> 500
<a>.<b>.<c>.<count2> 400
<a>.<b>.<c>.<count3> 0
<a>.<b>.<c>.<count4> 6001
<a>.<b>.<c>.<count5> 211
2112a
2110b
18000
2111b
<a>.<b>.<c>.<count1> 1250
<a>.<b>.<c>.<count2> 812
<a>.<b>.<c>.<count3> 1244
<a>.<b>.<c>.<count4> 0
<a>.<b>.<c>.<count6> 197
2112b
CDAE
2100
2115
2113
New count events:
<a>.<b>.<c>.<count6> 197
Removed count events:
<a>.<b>.<c>.<count5> 422
2114
Count events newly occurring:
<a>.<b>.<c>.<count3> 1244
Count events stopped occurring:
<a>.<b>.<c>.<count4> 0
2116
Count events with changes:
<a>.<b>.<c>.<count1> +25%

Fig. 21B

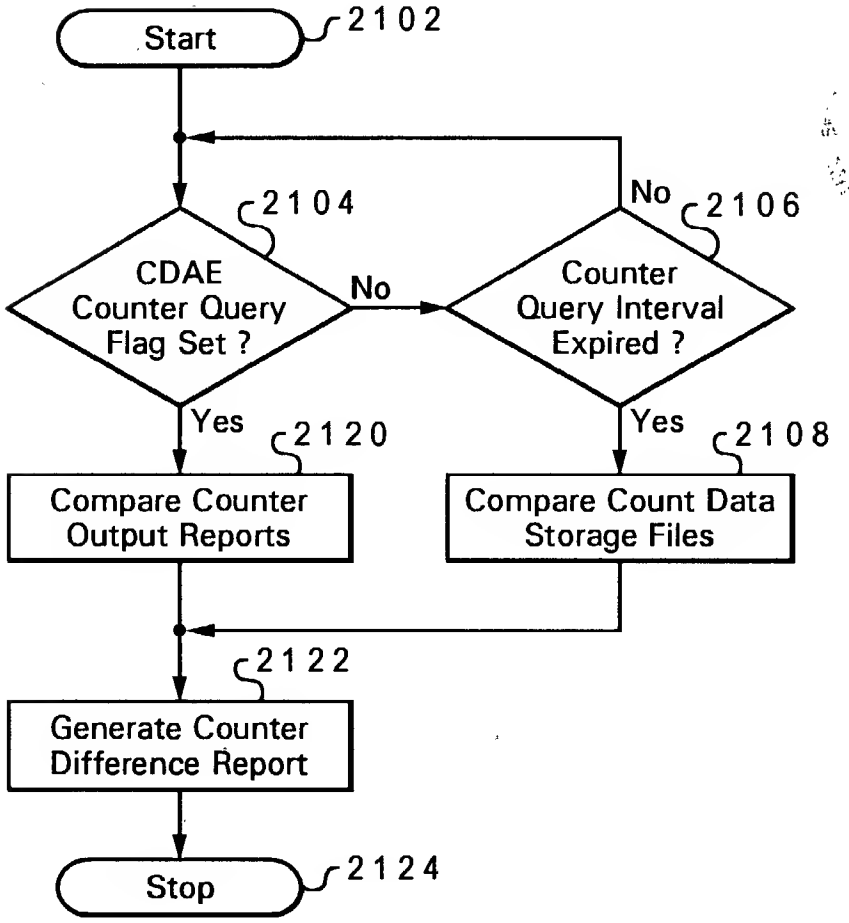

Start
2102
2104
CDAE Counter Query Flag Set ?
No
2106
Counter Query Interval Expired ?
No
Yes
2120
Compare Counter Output Reports
Yes
2108
Compare Count Data Storage Files
2122
Generate Counter Difference Report
Stop
2124

Fig. 21C

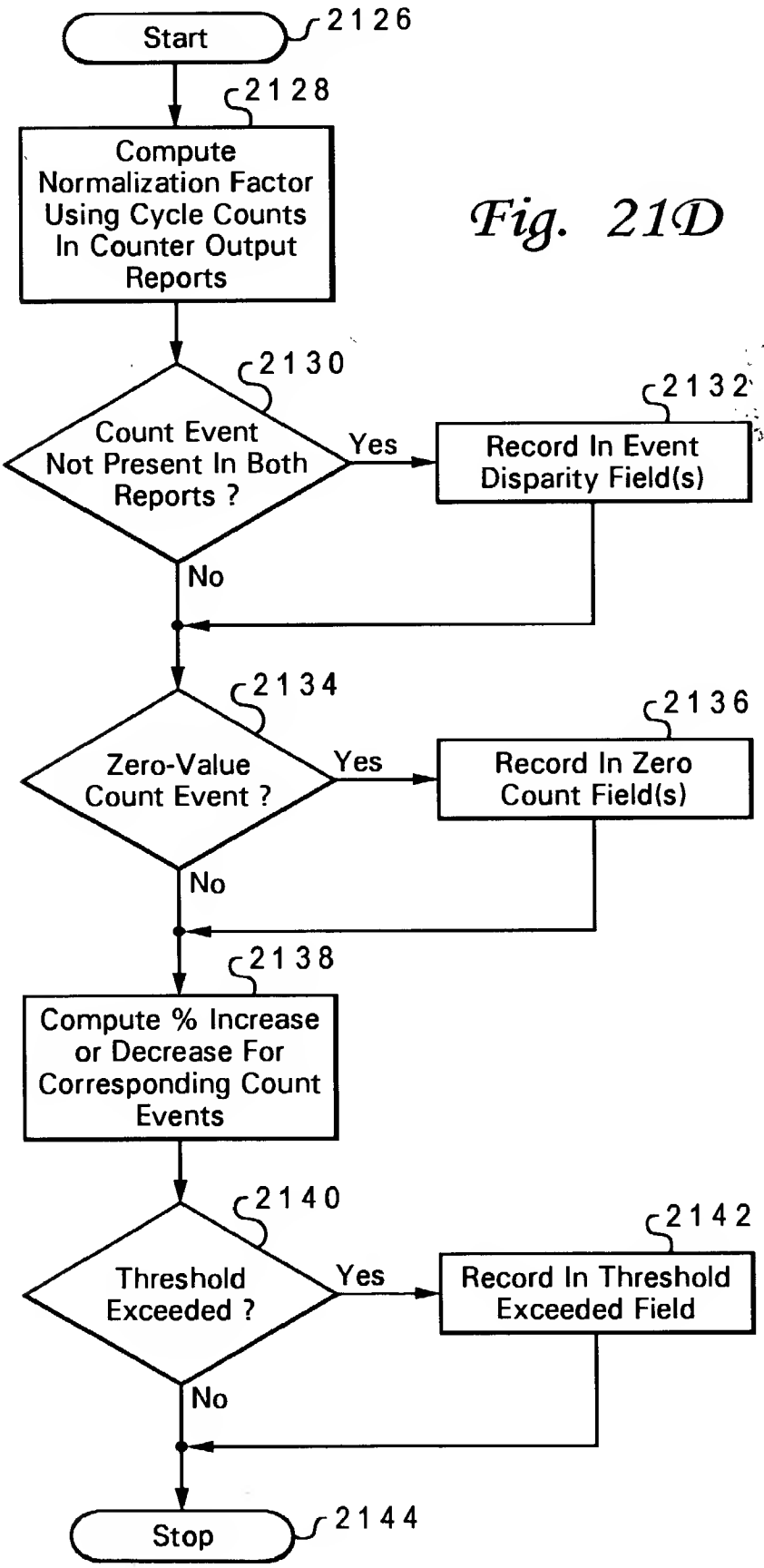

Start
2126
2128
Compute Normalization Factor Using Cycle Counts In Counter Output Reports
2130
Count Event Not Present In Both Reports ?
Yes
2132
Record In Event Disparity Field(s)
No
2134
Zero-Value Count Event ?
Yes
2136
Record In Zero Count Field(s)
No
2138
Compute % Increase or Decrease For Corresponding Count Events
2140
Threshold Exceeded ?
Yes
2142
Record In Threshold Exceeded Field
No
Stop
2144

Fig. 21D

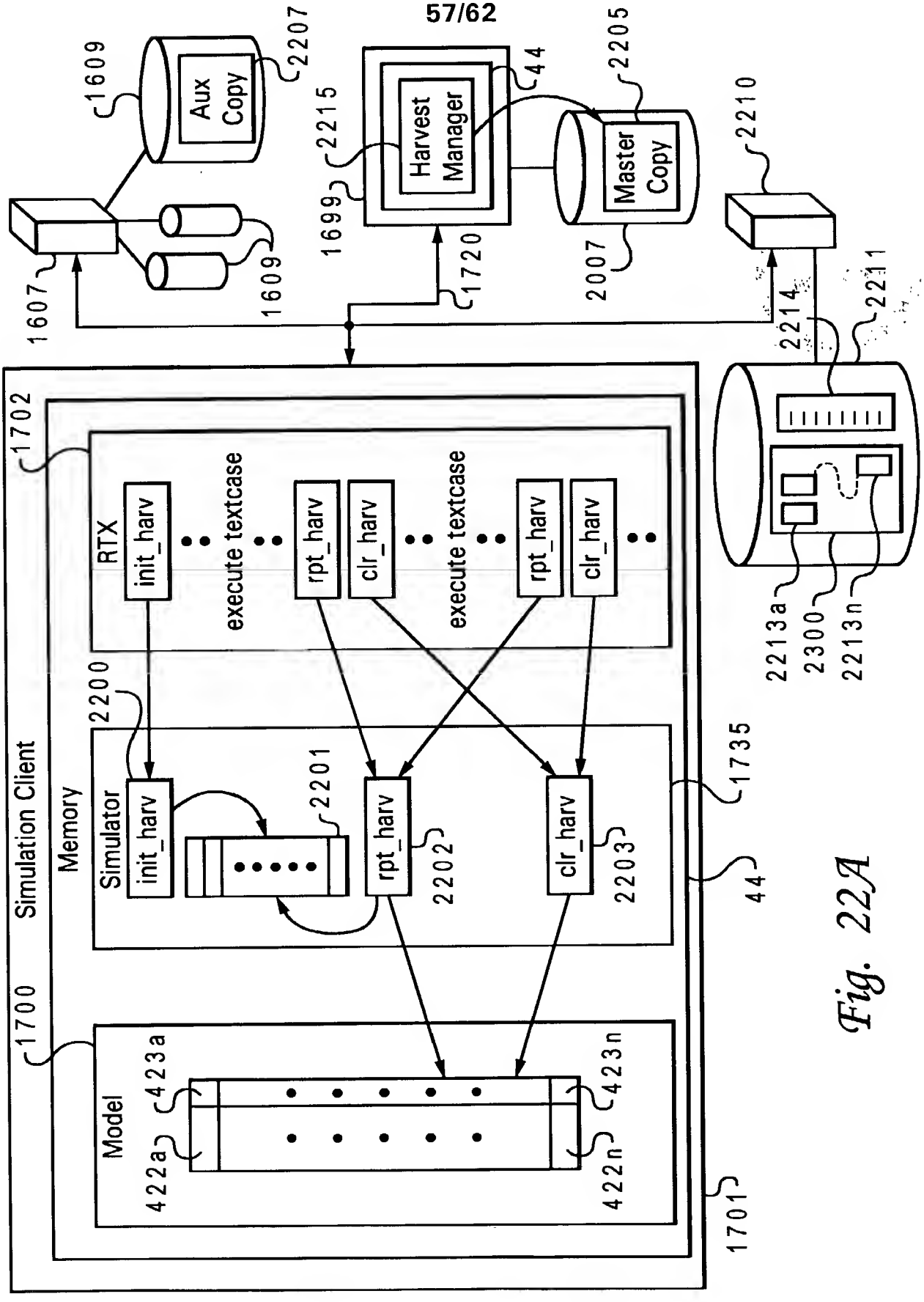

1609
Aux Copy
2207
2215
Harvest Manager
44
2205
Master Copy
2210
1699
1720
2007
1607
1609
2214
2211
2213a
2300
2213n
1702
RTX
init_harv
execute textcase
rpt_harv
clr_harv
execute textcase
rpt_harv
clr_harv
2200
Memory
Simulator
init_harv
2201
rpt_harv
2202
clr_harv
2203
1735
44
1700
Simulation Client
Model
423a
422a
423n
422n
1701

Fig. 22A

Fig. 22B

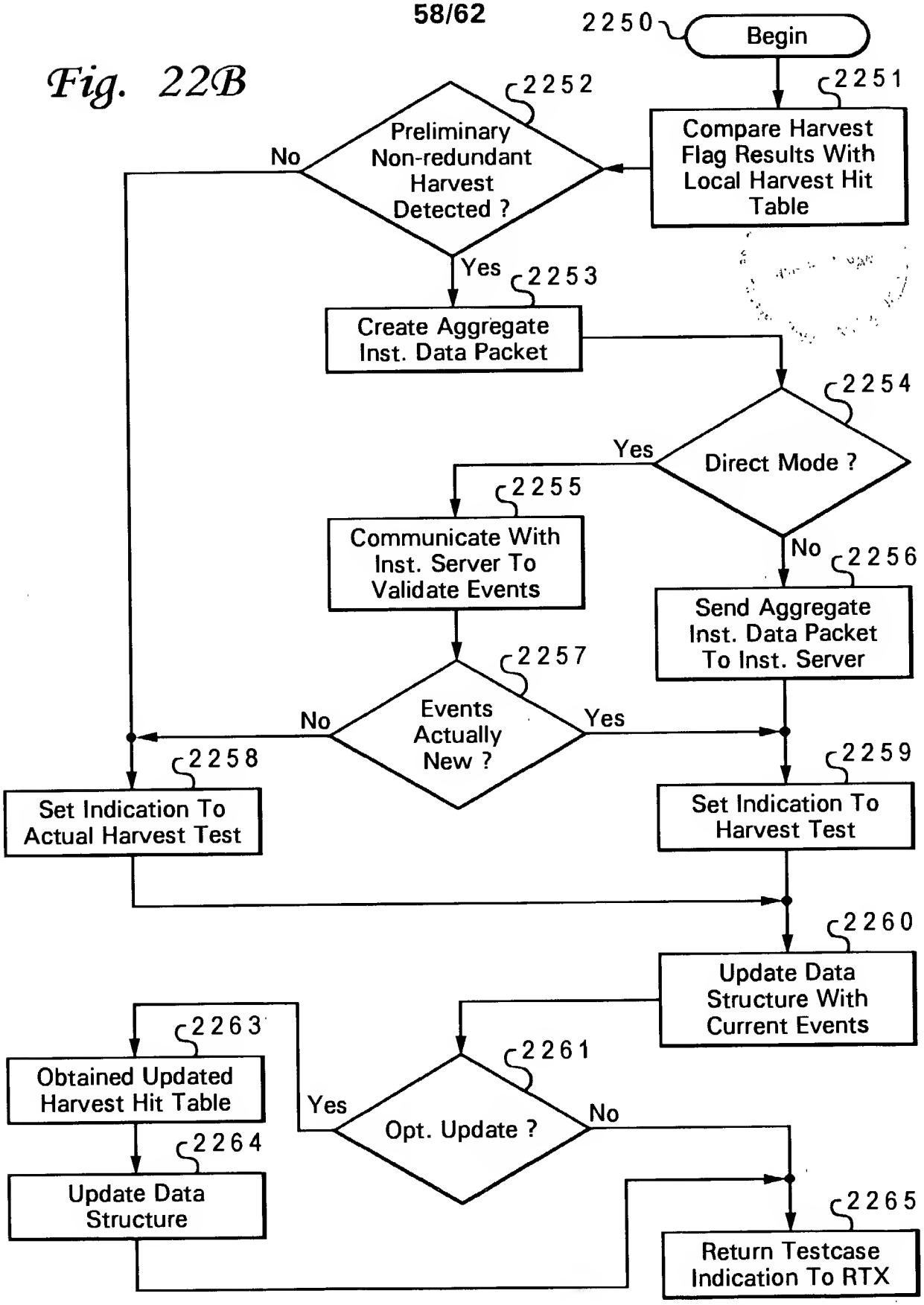
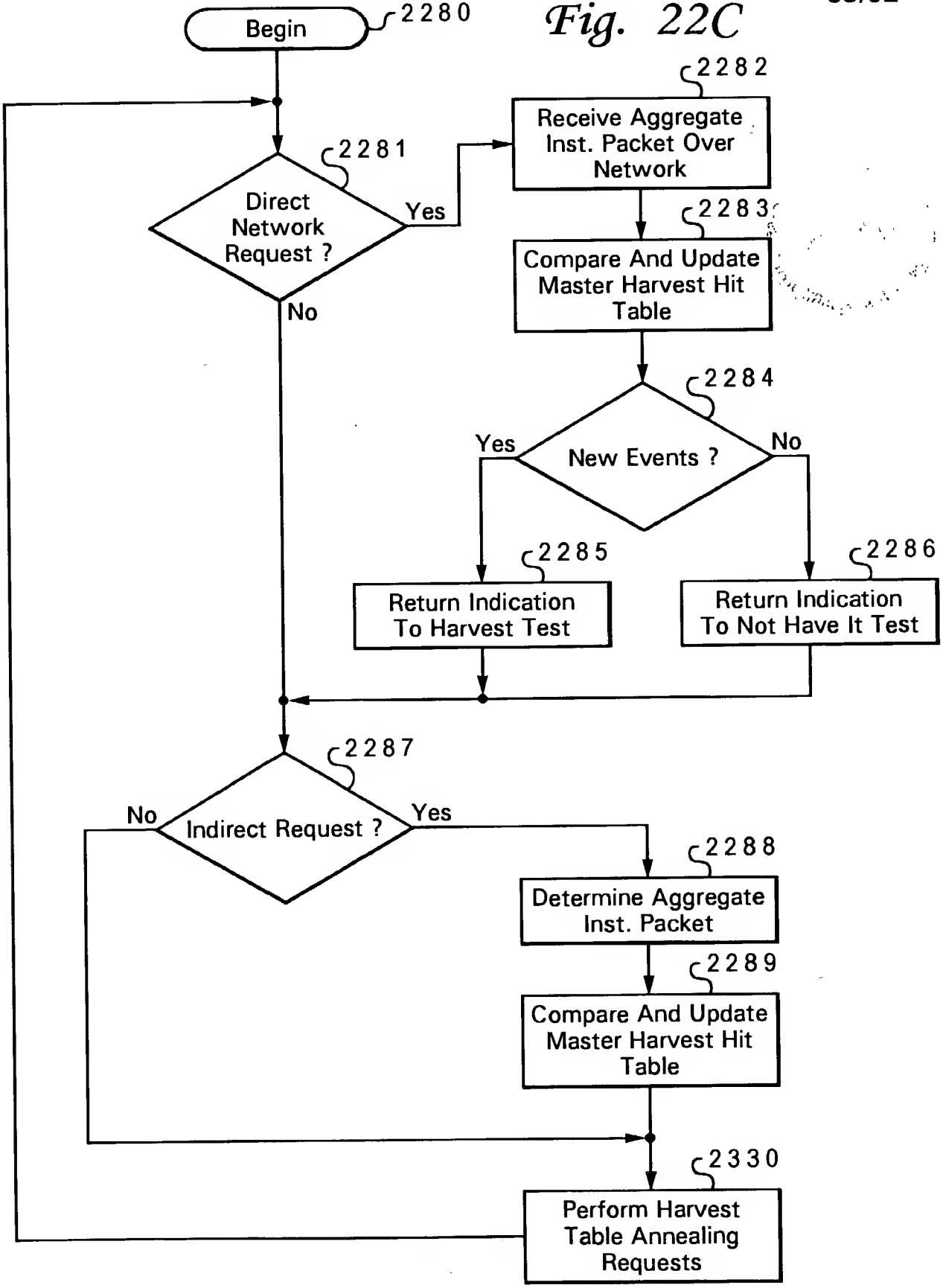

2250
Begin
2251
Compare Harvest Flag Results With Local Harvest Hit Table
2252
Preliminary Non-redundant Harvest Detected ?
No
Yes
2253
Create Aggregate Inst. Data Packet
2254
Direct Mode ?
Yes
No
2255
Communicate With Inst. Server To Validate Events
2256
Send Aggregate Inst. Data Packet To Inst. Server
2257
Events Actually New ?
No
Yes
2258
Set Indication To Actual Harvest Test
2259
Set Indication To Harvest Test
2260
Update Data Structure With Current Events
2261
Opt. Update ?
Yes
No
2263
Obtained Updated Harvest Hit Table
2264
Update Data Structure
2265
Return Testcase Indication To RTX

Fig. 22C


Begin
2280
2281
Direct Network Request ?
Yes
No
2282
Receive Aggregate Inst. Packet Over Network
2283
Compare And Update Master Harvest Hit Table
2284
New Events ?
Yes
No
2285
Return Indication To Harvest Test
2286
Return Indication To Not Have It Test
2287
Indirect Request ?
No
Yes
2288
Determine Aggregate Inst. Packet
2289
Compare And Update Master Harvest Hit Table
2330
Perform Harvest Table Annealing Requests

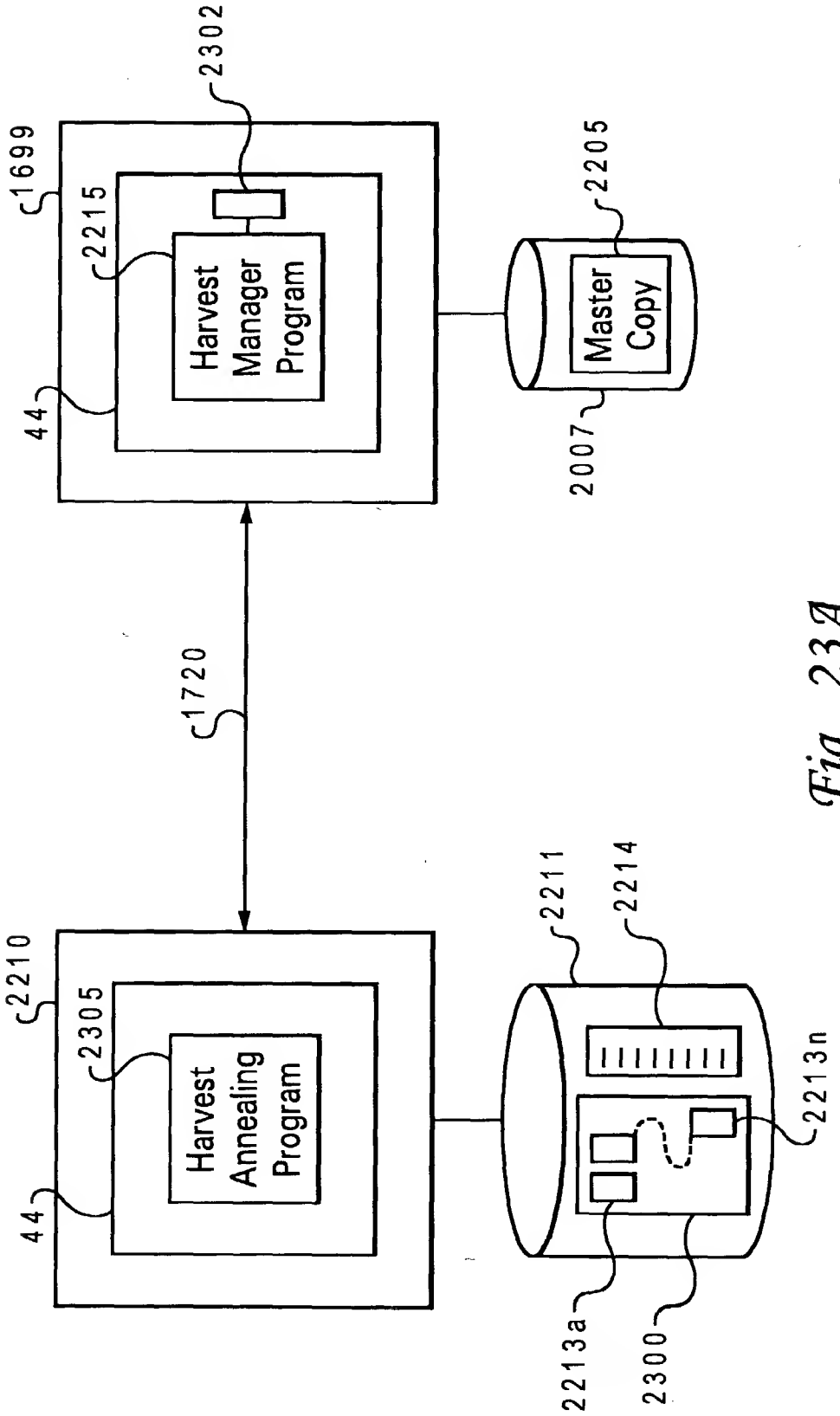

1699
2302
2215
Harvest Manager Program
44
2205
Master Copy
2007
1720
2210
2305
Harvest Annealing Program
44
2211
2214
2213n
2213a
2300

Fig. 23A

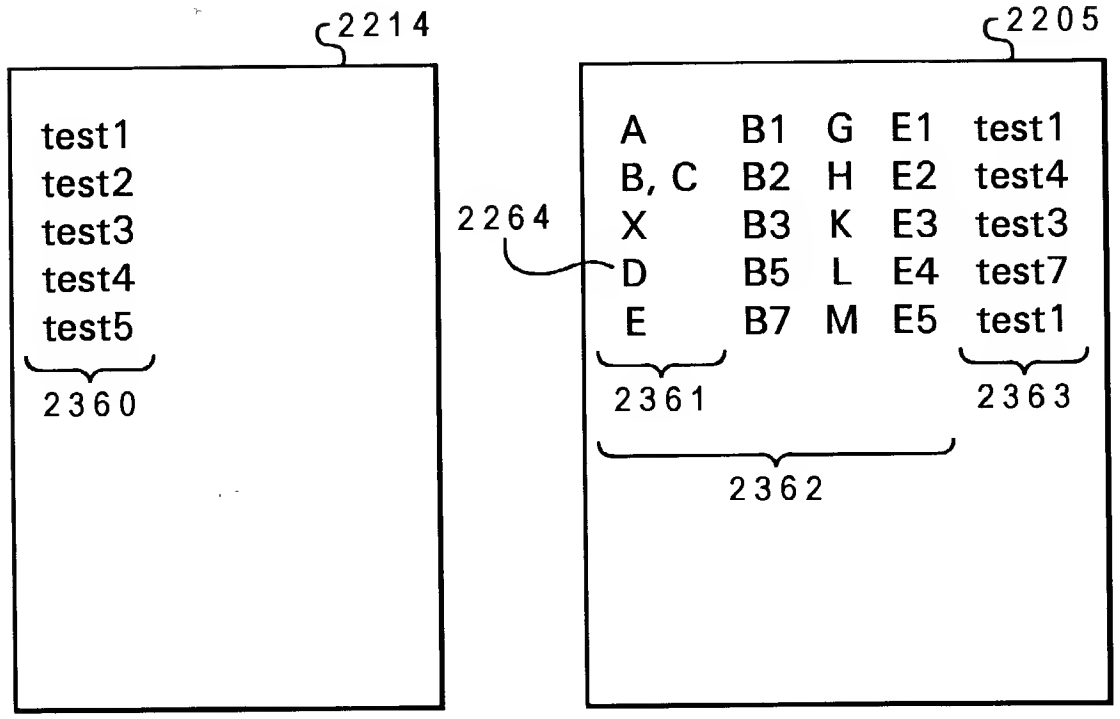

2214
2205
test1
test2
test3
test4
test5
2360
2264
A B1 G E1 test1
B, C B2 H E2 test4
X B3 K E3 test3
D B5 L E4 test7
E B7 M E5 test1
2361
2363
2362

Fig. 23B

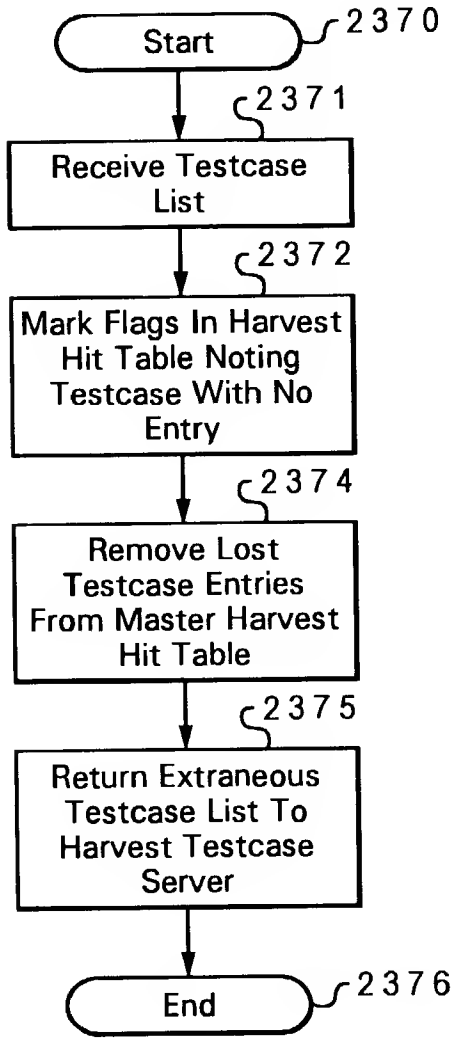

Start
2370
2371
Receive Testcase List
2372
Mark Flags In Harvest Hit Table Noting Testcase With No Entry
2374
Remove Lost Testcase Entries From Master Harvest Hit Table
2375
Return Extraneous Testcase List To Harvest Testcase Server
End
2376

Fig. 23C